

# Fundamental Tidy Data Science in R

Muhammad Aswan Syahputra  
R Academy, Telkom University, Bandung  
26 - 27 July 2019



- Sensory Scientist @ [Sensolution.ID](#)
- Using R for 4+ years, keen on Data Carpentry
- Initiator of [Komunitas R Indonesia](#)
- 📦: sensehubr, nusandata, bandungjuara, prakiraan, etc
- 💻: sensehub, thermostats, aquastats, bcrp, bandungjuara, etc

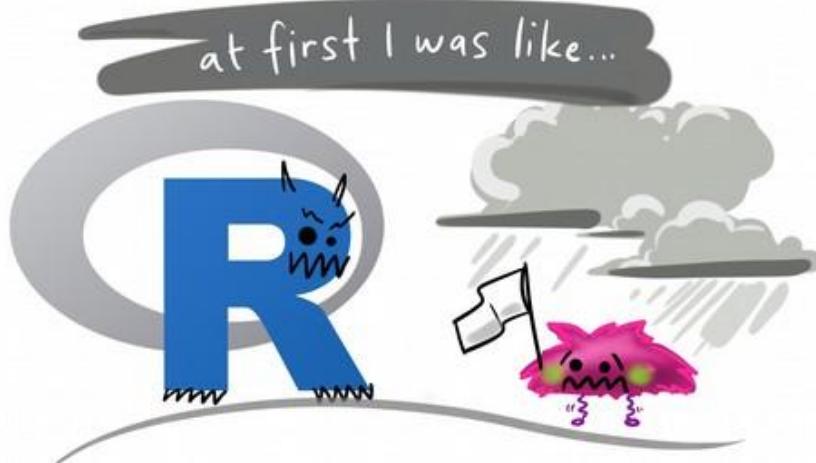


aswansyahputra

# Know your neighbour!

- Who are you?
- What you do with data?
- How would you describe your experience with R?

# Our goal

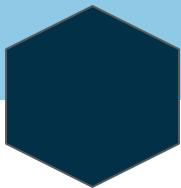


...but now it's like...

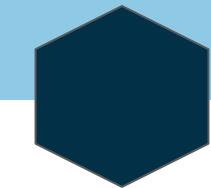


Artwork by @allison\_horst

# Meet the assistants

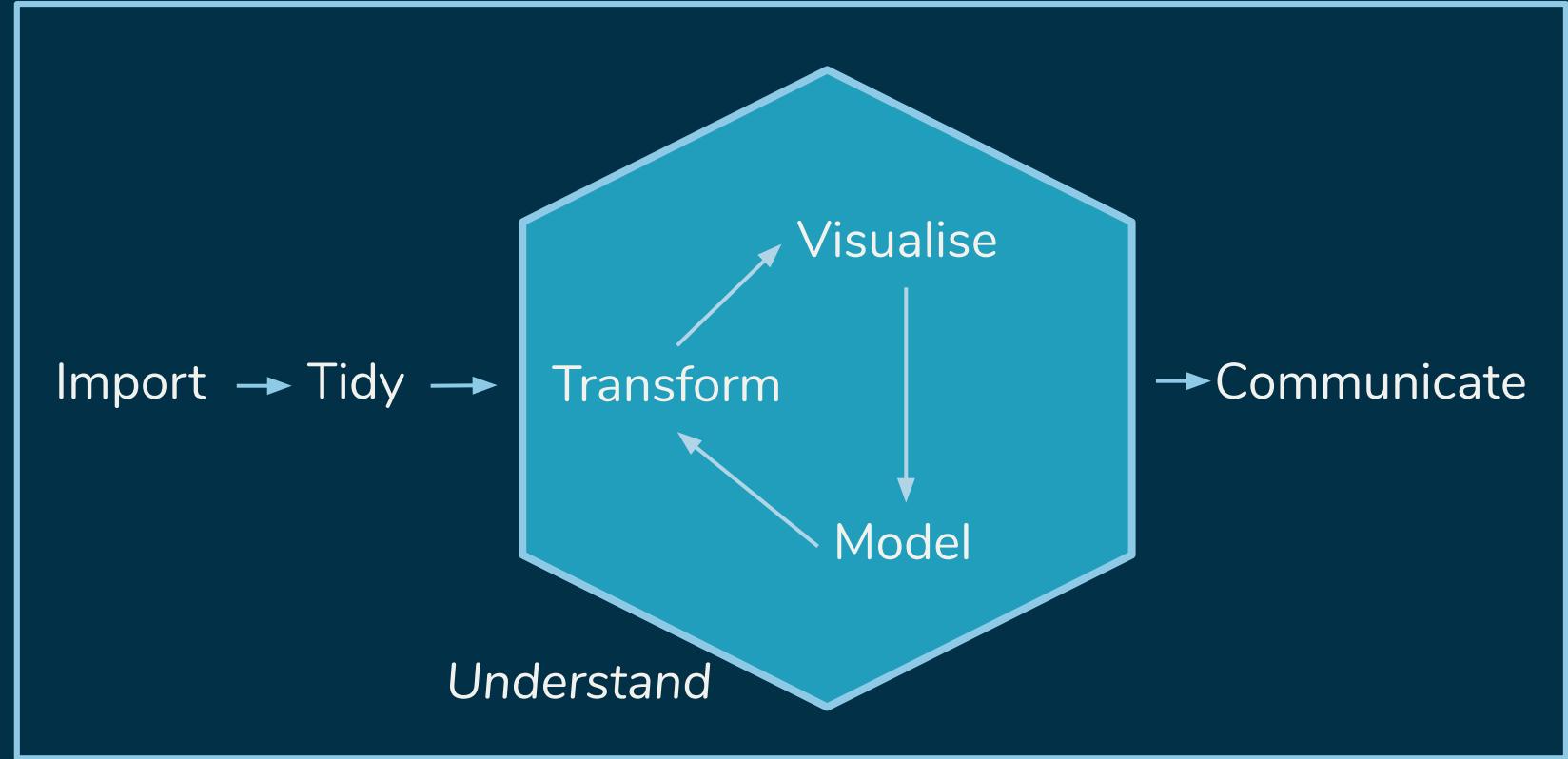


Naviz

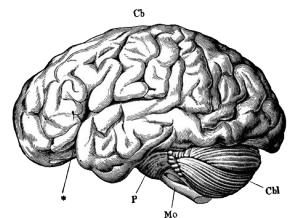


William

Data science is the art  
of turning raw data into  
understanding



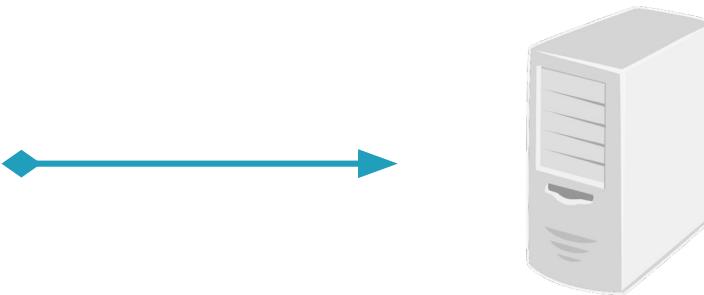
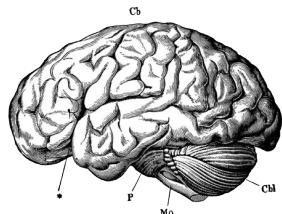
*Program*



Human thought



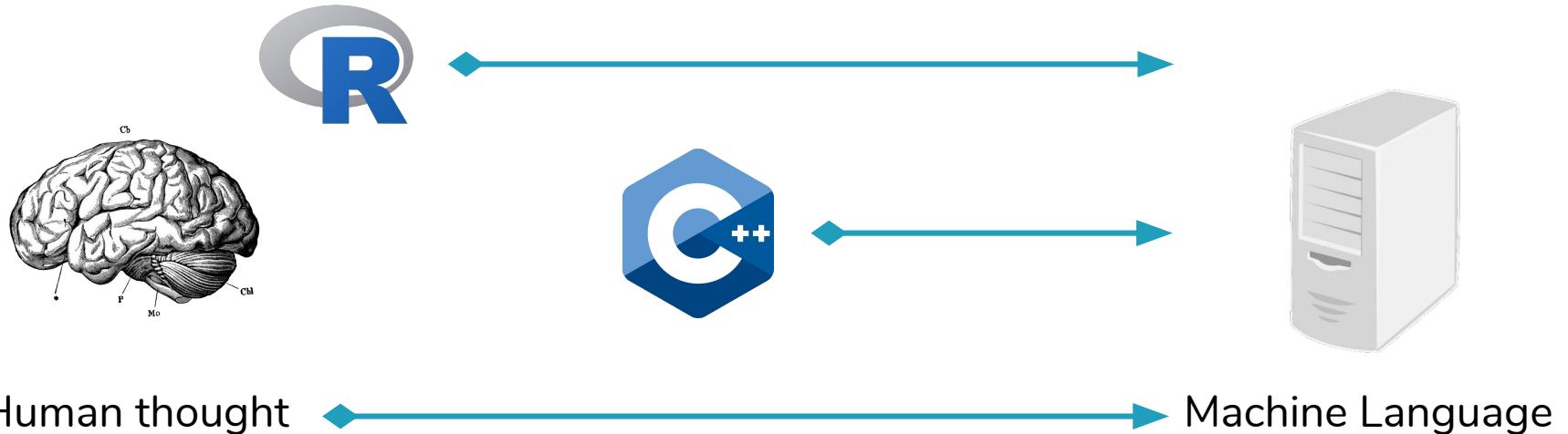
Machine Language



Human thought

Machine Language



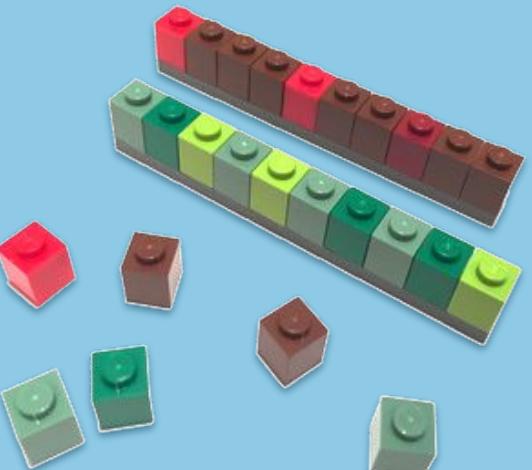


# Data structure

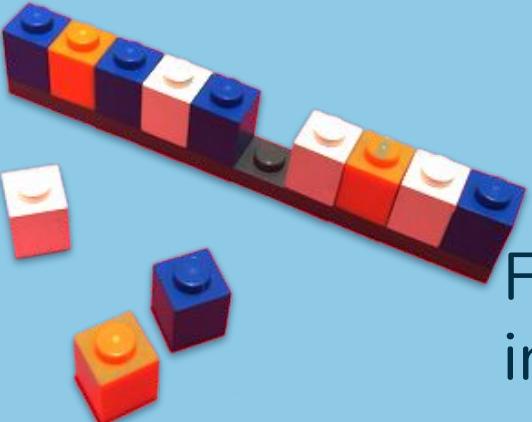
Dimension	Homogenous	Heterogenous
1d	Atomic vectors	List
2d	Matrix	Dataframe
nd	Array	

# Data structure

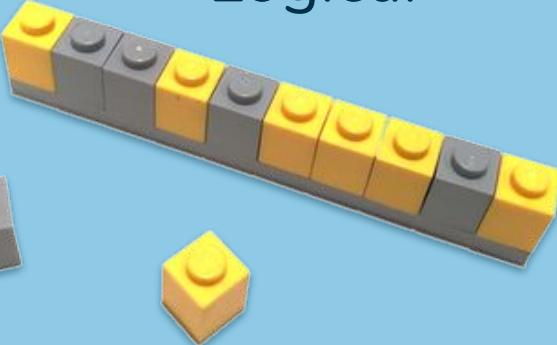
Dimension	Homogenous	Heterogenous
1d	Atomic vectors	List
2d	Matrix	Dataframe
nd	Array	



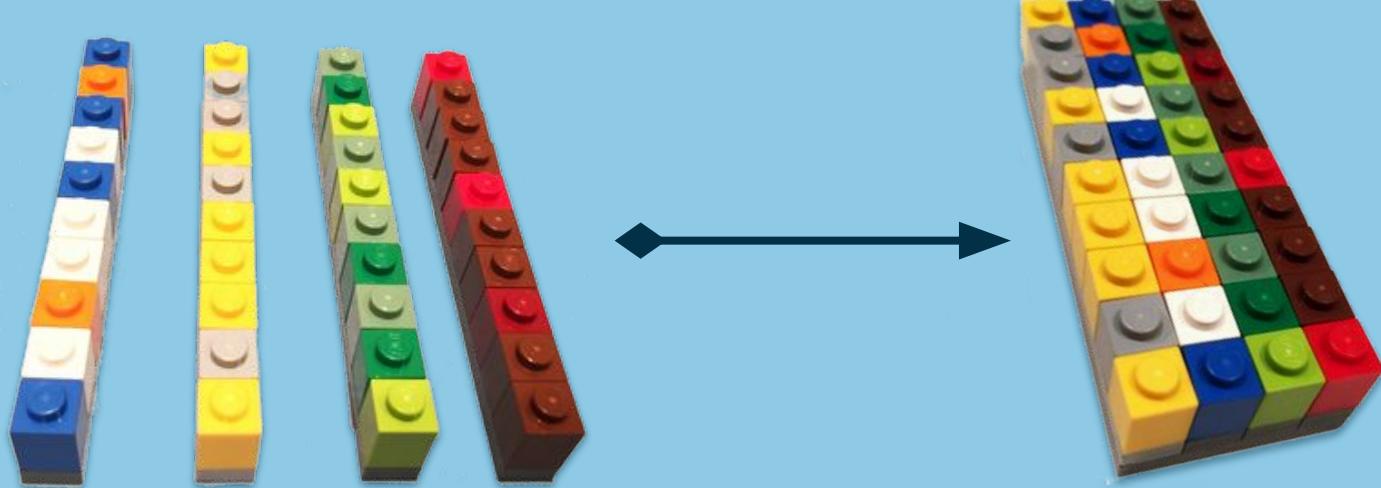
Integer, Double, Character



Factor (basically  
integer with class)



Logical



Vectors of same length

Dataframe

How do we process  
the data?

```
function(arg1, arg2, arg3, ...)
```

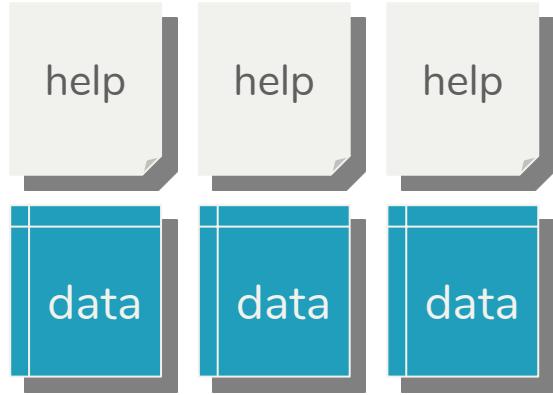
```
graph TD; A[change_the_env(...)] --> B(function); C[calculate_value(...)] --> B
```

change\_the\_env(...)

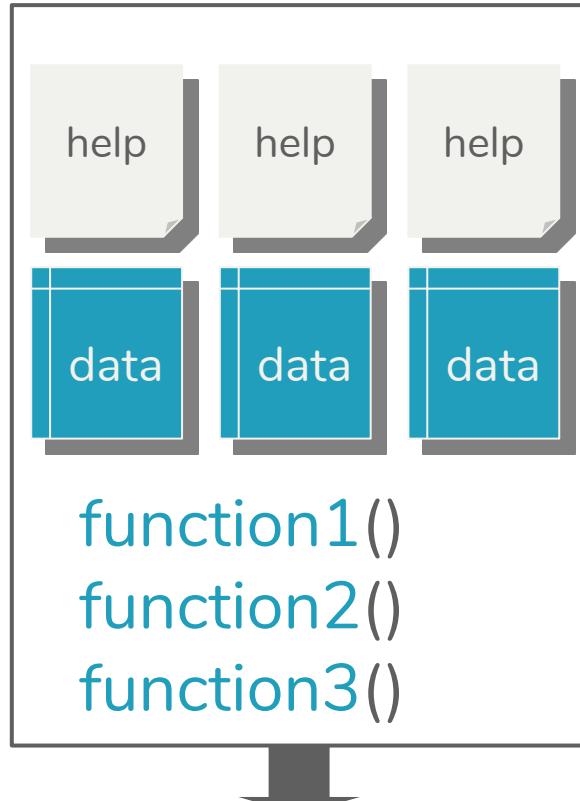
calculate\_value(...)

assign. <- , =, ->

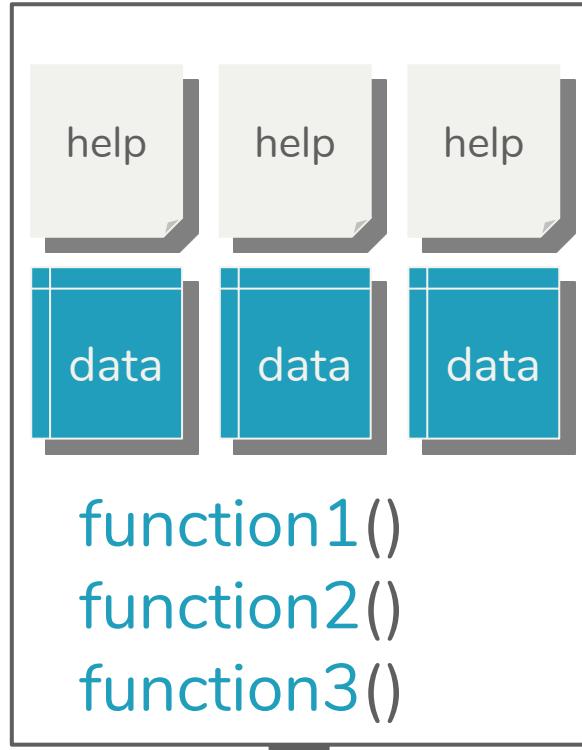
- arguments are contexts of a function
- arguments are matched by name, or
- arguments are matched by position, **be careful!**



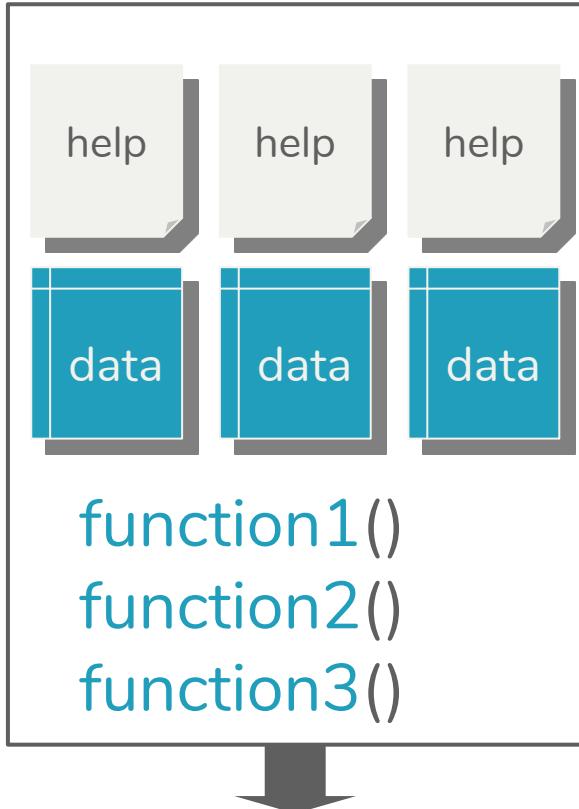
`function1()`  
`function2()`  
`function3()`



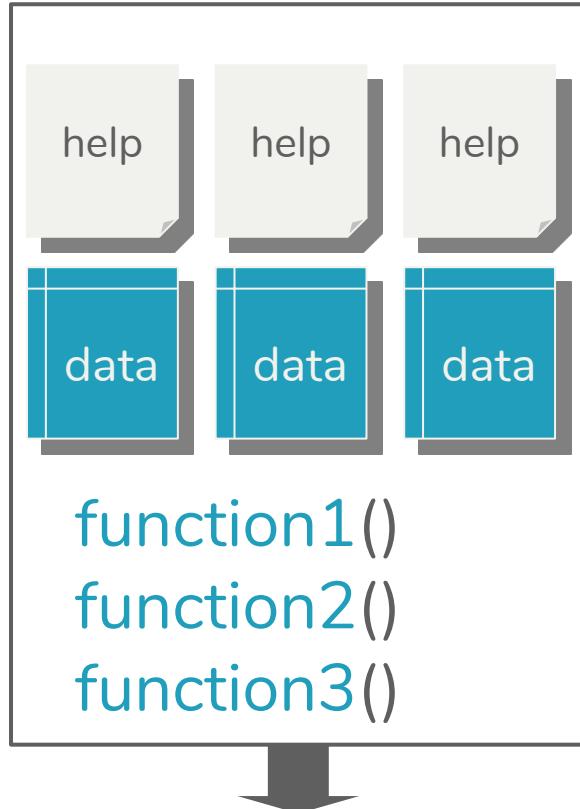
stats, graphics, grDevices, utils,  
datasets, methods, base



stats, graphics, grDevices, utils,  
datasets, methods, base



R packages



# CRAN Task Views

<https://cran.r-project.org/web/views/>

The screenshot shows a dark-themed web browser window displaying the CRAN Task Views page. The title bar reads "The Comprehensive R Arc" and the URL is "https://cran.r-project.org". The page features the R logo and a navigation menu on the left with links like CRAN Mirrors, What's new?, Task Views, Search, About R, R Homepage, and The R Journal. The main content area is titled "CRAN Task Views" and discusses the purpose of task views. It includes a bulleted list of instructions for installing and maintaining task views, and a "Topics" section listing various R packages categorized by topic.

**CRAN Task Views**

CRAN task views aim to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic. They give a brief overview of the included packages and can be automatically installed using the [ctv](#) package. The views are intended to have a sharp focus so that it is sufficiently clear which packages should be included (or excluded) - and they are *not* meant to endorse the "best" packages for a given task.

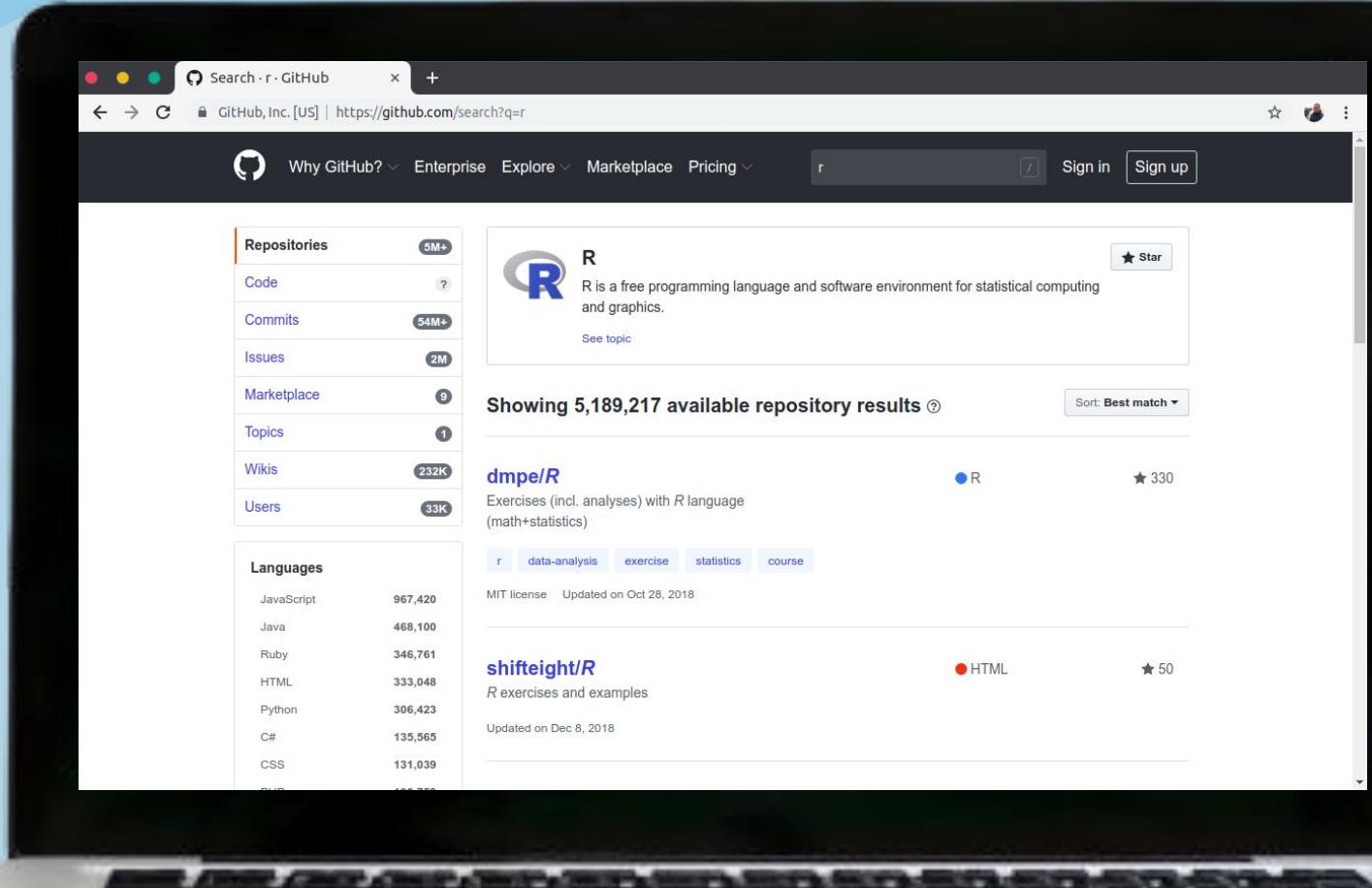
- To automatically install the views, the [ctv](#) package needs to be installed, e.g., via`install.packages("ctv")`and then the views can be installed via `install.views` or `update.views` (where the latter only installs those packages are not installed and up-to-date), e.g.,`ctv::install.views("Econometrics")``ctv::update.views("Econometrics")`
- The task views are maintained by volunteers. You can help them by suggesting packages that should be included in their task views. The contact e-mail addresses are listed on the individual task view pages.
- For general concerns regarding task views contact the [ctv](#) package maintainer.

**Topics**

<a href="#">Bayesian</a>	Bayesian Inference
<a href="#">ChemPhys</a>	Chemometrics and Computational Physics
<a href="#">ClinicalTrials</a>	Clinical Trial Design, Monitoring, and Analysis
<a href="#">Cluster</a>	Cluster Analysis & Finite Mixture Models
<a href="#">Databases</a>	Databases with R
<a href="#">DifferentialEquations</a>	Differential Equations
<a href="#">Distributions</a>	Probability Distributions
<a href="#">Econometrics</a>	Econometrics
<a href="#">Environmetrics</a>	Analysis of Ecological and Environmental Data
<a href="#">ExperimentalDesign</a>	Design of Experiments (DoE) & Analysis of Experimental Data
<a href="#">ExtremeValue</a>	Extreme Value Analysis
<a href="#">Finance</a>	Empirical Finance
<a href="#">FunctionalData</a>	Functional Data Analysis
<a href="#">Genetics</a>	Statistical Genetics
<a href="#">Graphics</a>	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

# GitHub

[https://github.com/  
search?q=r](https://github.com/search?q=r)



The screenshot shows a GitHub search results page for the query "r". The top navigation bar includes links for "Why GitHub?", "Enterprise", "Explore", "Marketplace", "Pricing", and user authentication options ("Sign in" and "Sign up"). The search bar contains the query "r".

**Repositories** (5M+)

- Code
- Commits (54M+)
- Issues (2M)
- Marketplace (9)
- Topics (1)
- Wikis (232K)
- Users (33K)

**R** R is a free programming language and software environment for statistical computing and graphics.

See topic

**Showing 5,189,217 available repository results** Sort: Best match

**dmpe/R** R ★ 330 Exercises (incl. analyses) with R language (math+statistics)

JavaScript 967,420 Java 468,100 Ruby 346,761 HTML 333,048 Python 306,423 C# 135,565 CSS 131,039 PHP 106,750

MIT license Updated on Oct 28, 2018

**shifteight/R** HTML ★ 50 R exercises and examples

Updated on Dec 8, 2018

## Installing package, only once

```
install.packages("pkg") # from CRAN/local  
remotes::install_github("user/pkg") # from GitHub  
remotes::install_bioc("repo") # from Bioconductor
```

## Loading package, once in every session

```
library(pkg)  
pacman::p_load(pkg) # load or install if not  
available
```

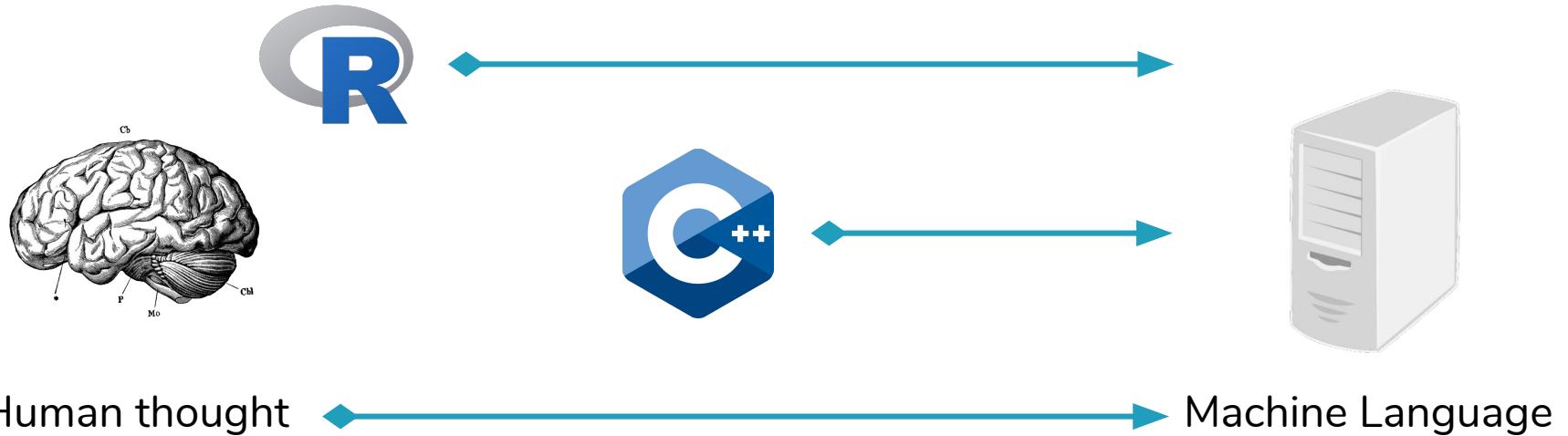
A lot of R packages  
to use! :D

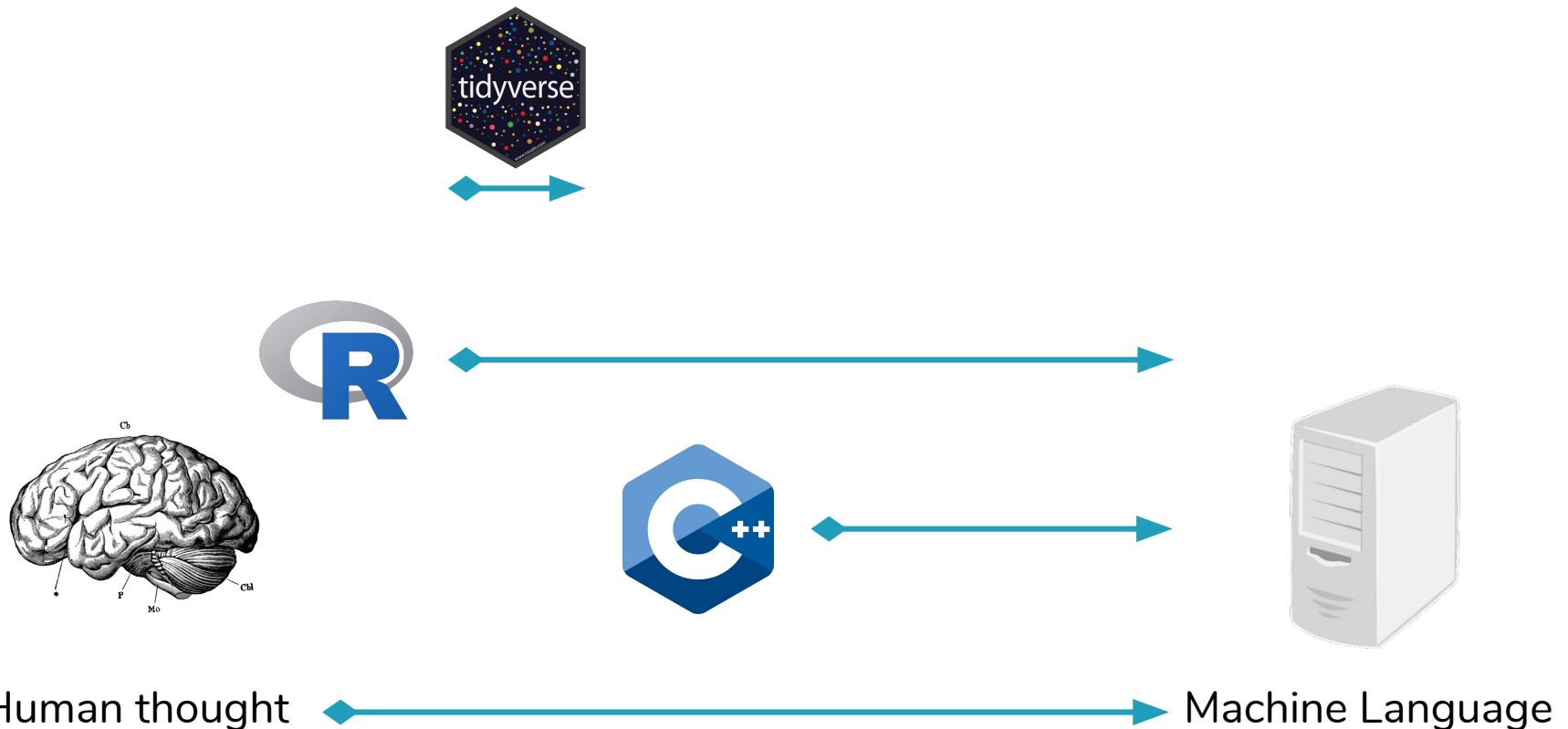
A lot of R packages  
to use! :(

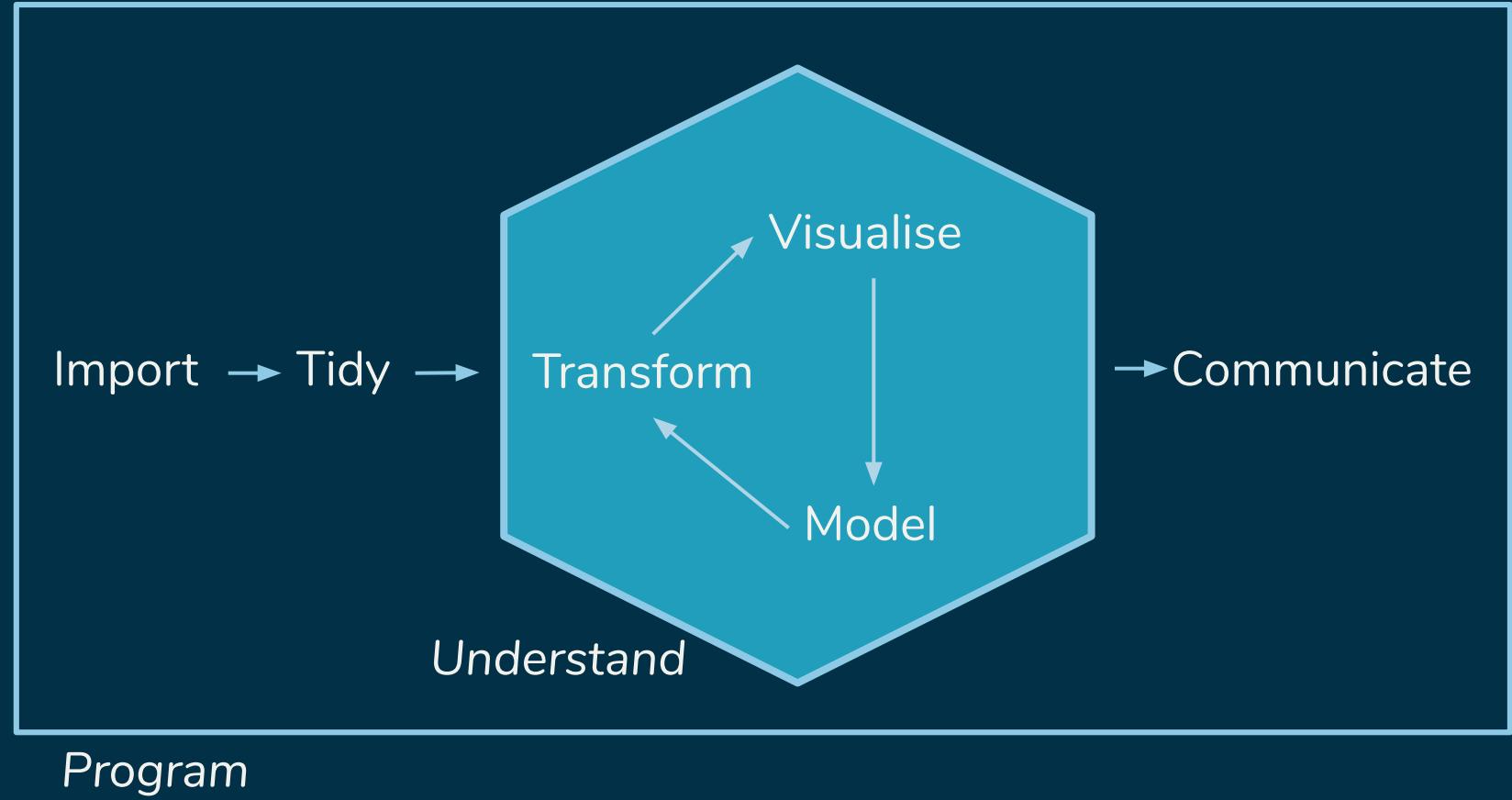


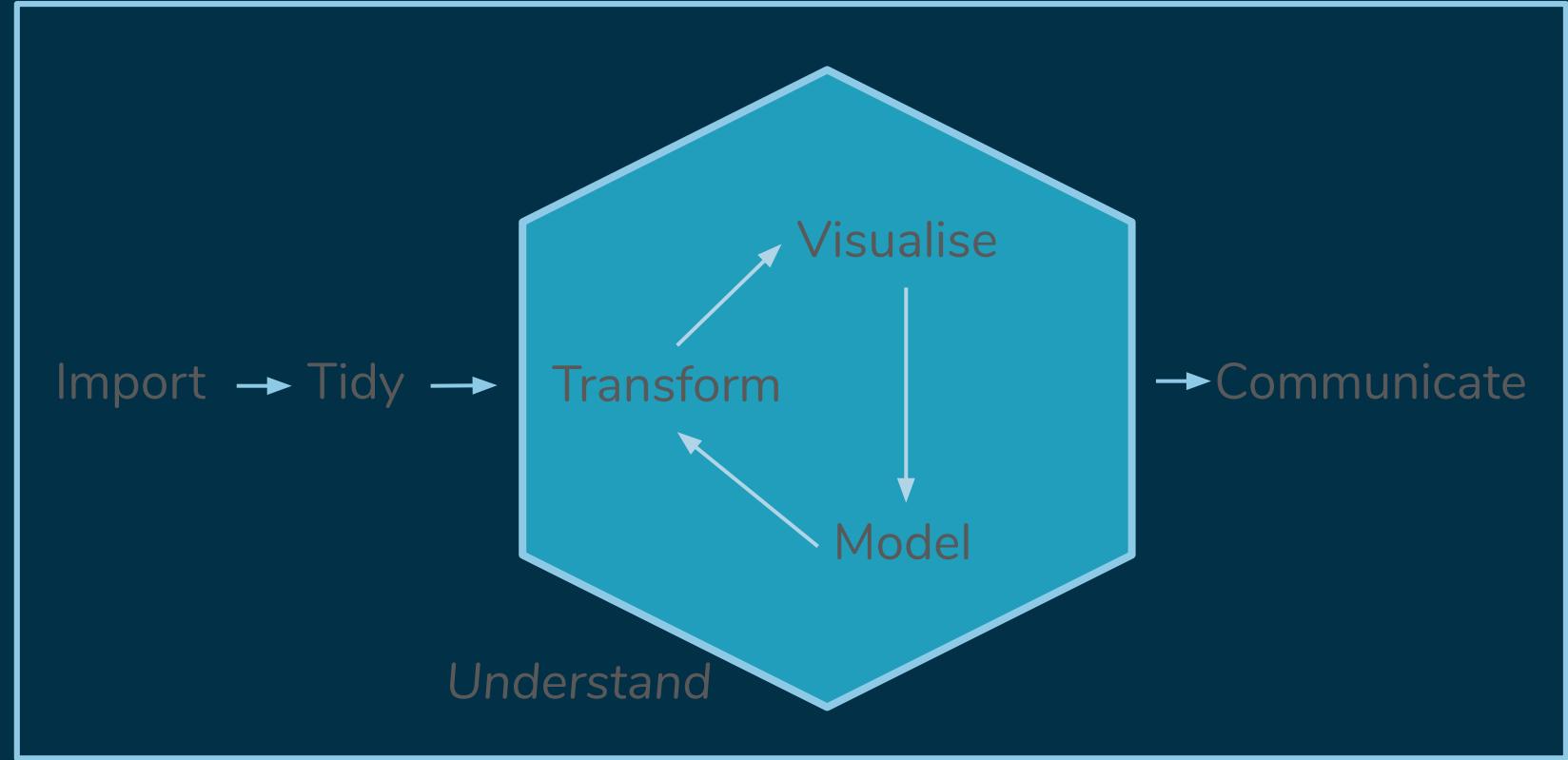


The tidyverse is an opinionated **collection of R packages** designed for **data science**. All packages **share** an underlying **design philosophy, grammar, and data structures**.









**Program**



**tidymodels**

**tidyverts**

**tidygraph**

**tidycode**

**tidync**

**tidycensus**

**tidytext**

**tidygenomics**

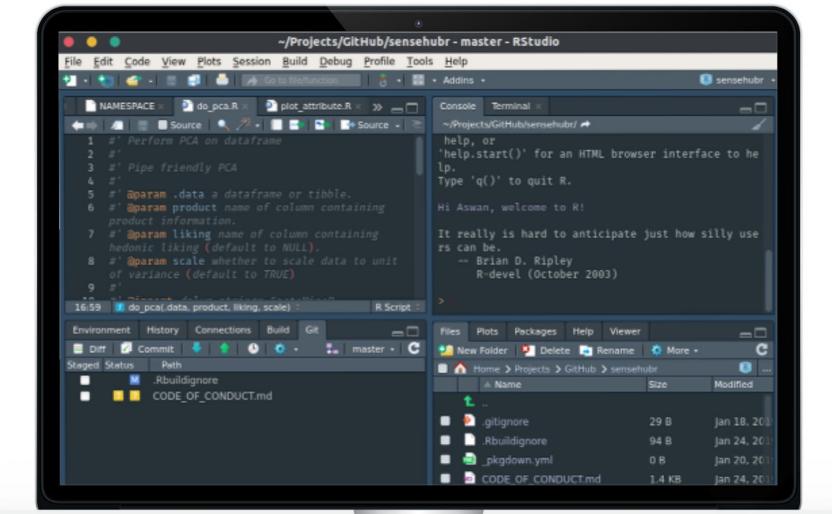


Where to  
code?

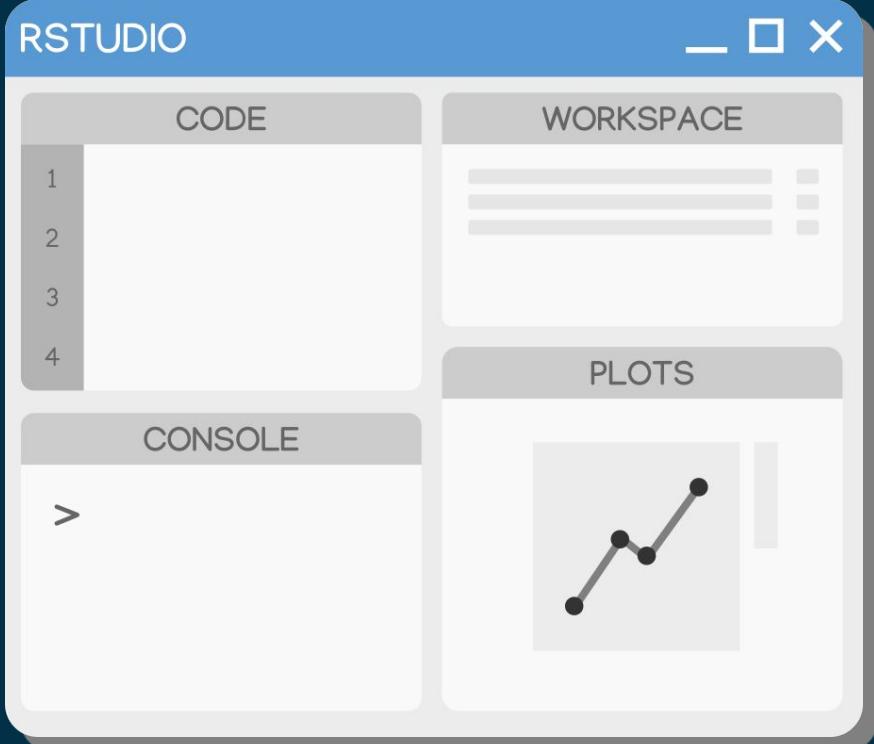


## Main features:

- Console
- Syntax-highlighting editor
- Tools for plotting, history, debugging and workspace management



[rstudio.com/products/rstudio/download/](http://rstudio.com/products/rstudio/download/)



- **Tab**, autocomplete & path navigation
- **Alt + -**, for assignment operator <->
- **Ctrl + Shift + M**, for pipe operator %>%
- **Ctrl + Enter**, run current line code/example on help page
- **Ctrl + Up**, search for code history on console or editor pane
- **Alt + Up/Down**, move code to above or below
- **Alt + Shift + Up/Down**, copy code to above or below
- **Ctrl + D**, delete current line
- **Ctrl + Shift + F10**, restart R session
- **Ctrl + Alt + B**, run code up to current line

Cheatsheet: <https://github.com/rstudio/cheatsheets/raw/master/rstudio-ide.pdf>

# Rmarkdown

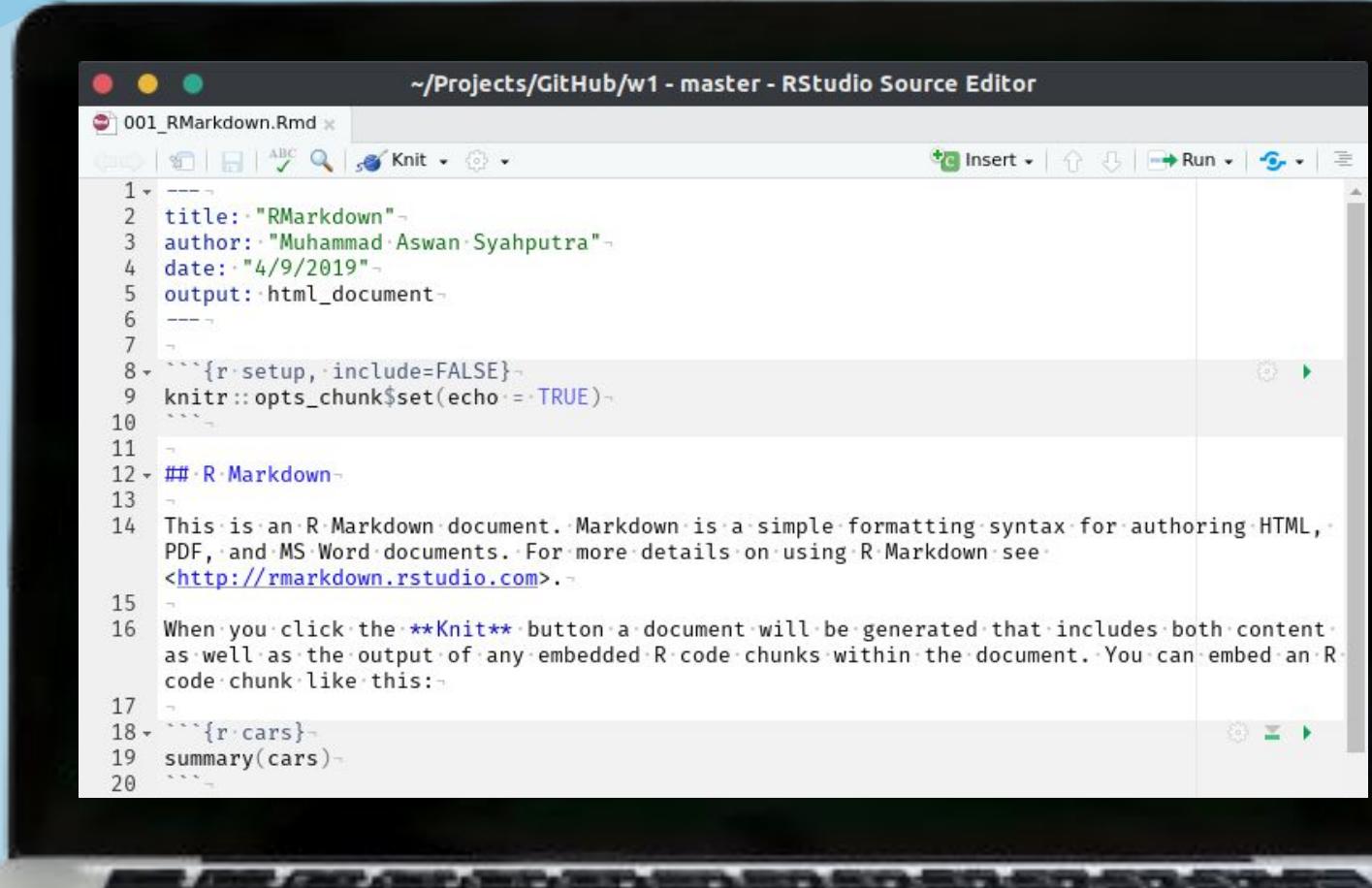
TEXT. CODE. OUTPUT.  
(GET IT TOGETHER, PEOPLE.)



Artwork by @allison\_horst

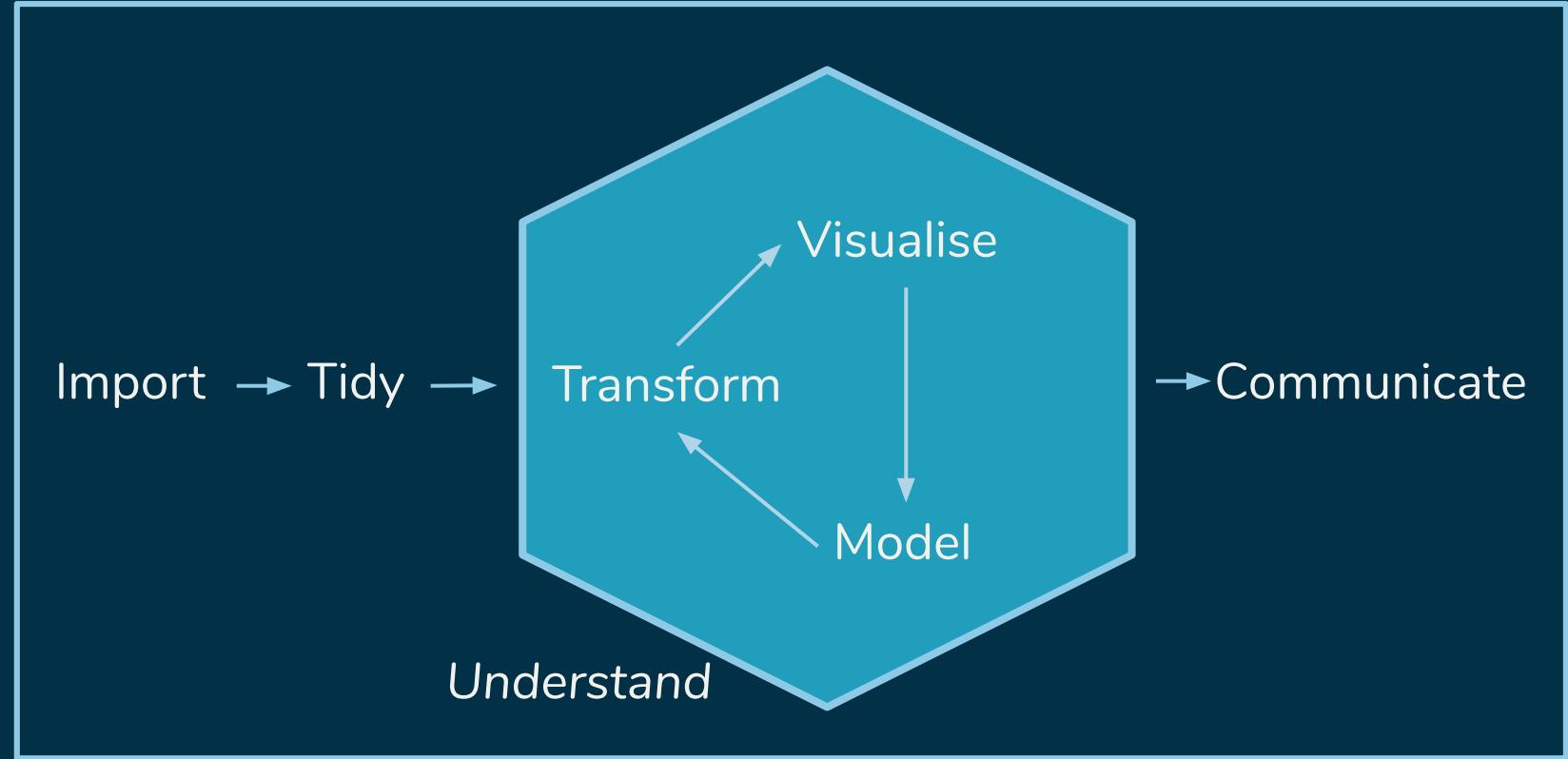
# R Markdown

A document format for authoring data science project

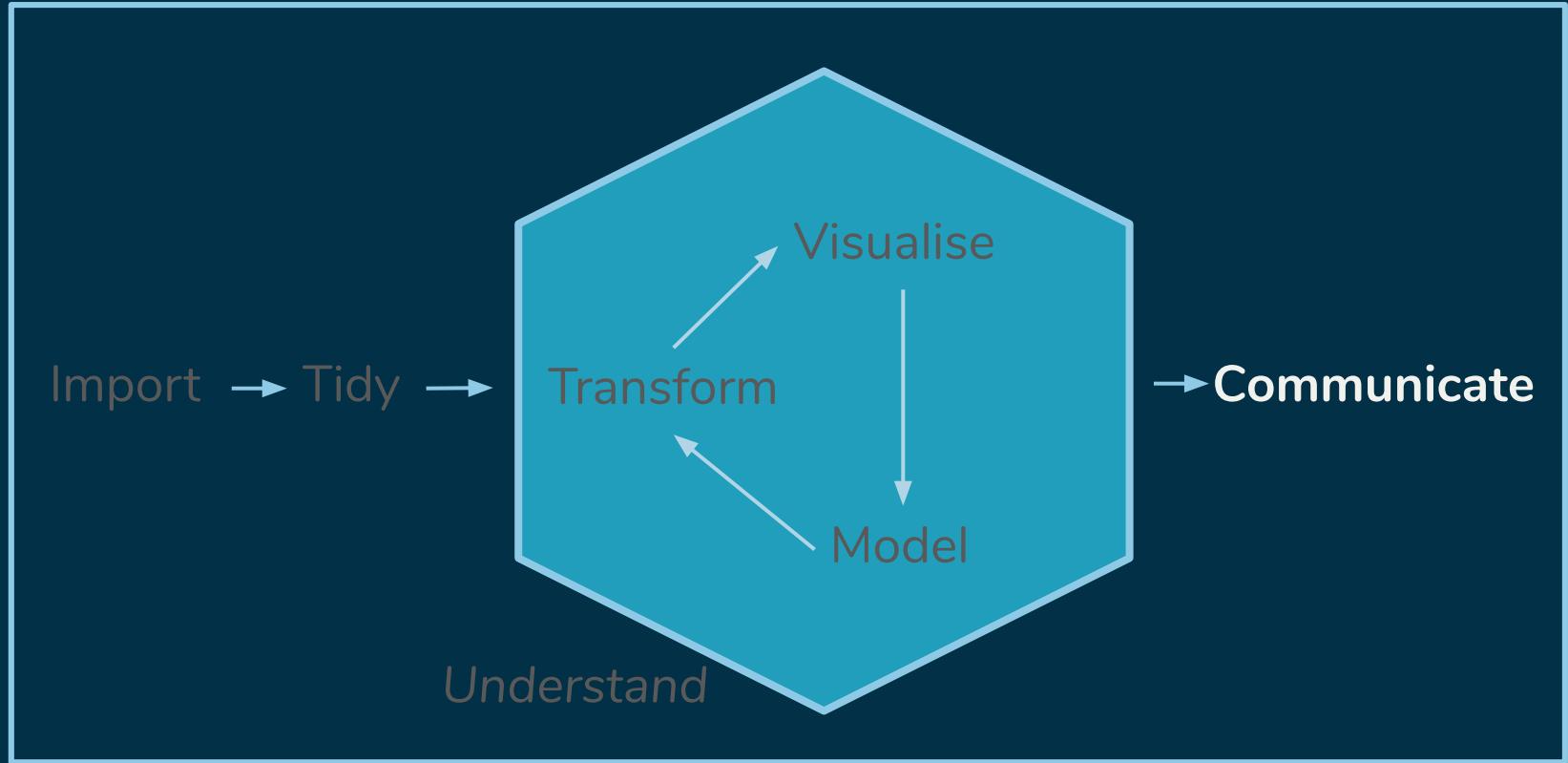


The screenshot shows the RStudio Source Editor interface. The title bar reads " ~/Projects/GitHub/w1 - master - RStudio Source Editor". The main window displays an R Markdown file named "001\_RMarkdown.Rmd". The code is as follows:

```
1 ---  
2 title: "RMarkdown"  
3 author: "Muhammad Aswan Syahputra"  
4 date: "4/9/2019"  
5 output: html_document  
---  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ````  
11 ## R Markdown  
12 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see  
http://rmarkdown.rstudio.com.  
15  
16 When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:  
17  
18 ```{r cars}  
19 summary(cars)  
20 ````
```



*Program*



*Program*

- Use script (R Markdown or R Script), try to avoid console
- Use Projects, not setwd(...) in script
- Set stringsAsFactor = FALSE, but not in the .Rprofile
- Ctrl+Shift+F10 and Ctrl+Alt+B to clean up,
- not rm(list=ls())
- Learn the handy shortcuts
- Do not save and load .Rdata
- Use version control system: git!

Reading: [happygitwithr.com](http://happygitwithr.com)



Download: [git-scm.com](http://git-scm.com)

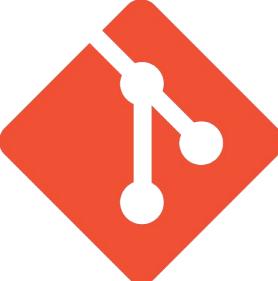


With great codes,  
comes great bugs!  
- (not) Uncle Ben

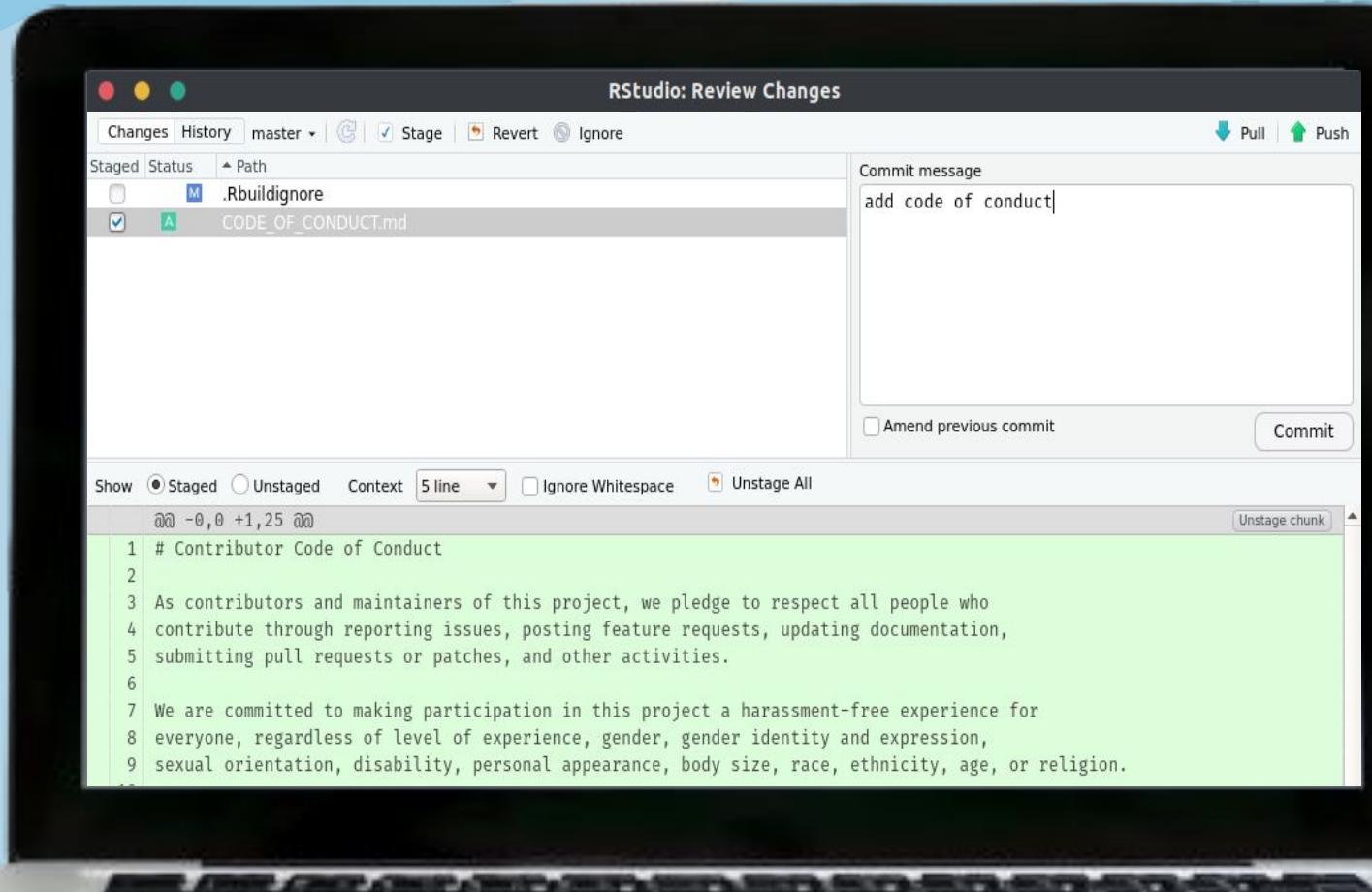
Store and share! Why sharing your work? Motivation [here](#).

- git clone <https://github.com/user/repo>
- Do some works!
- git add file.R or git add .
- git commit -m “what you have done”
- git push origin master
- Repeat: work, git add, git commit, git push



- 
- The Git logo consists of the word "git" in a large, dark brown, sans-serif font. To the left of the "g", there is a red diamond shape containing a white icon of two nodes connected by a line, representing a commit graph.
- git init
  - git remote add origin <https://github.com/user/repo>
  - Do some works!
  - git add file.R or git add .
  - git commit -m “what you have done”
  - git push -u origin master #use -u only once
  - Repeat: work, git add, git commit, git push

# It is available in RStudio!



# Let's get them!

- . Download R at  
[cran.r-project.org](http://cran.r-project.org)
- . Download RStudio at  
[rstudio.com](http://rstudio.com)
- . Download git  
[git-scm.com](http://git-scm.com)

# How to set?

- Create account at [github.com](https://github.com)
- In Rstudio, Tools – Global Options – Git/SVN – Browse git executable:
  - Windows:
    - **C:/Program Files/Git/bin/git.exe**
  - UNIX/UNIX-like:
    - **/usr/bin/git**

# How to set?

- Open git bash/shell/terminal
- Run:

```
git config --global user.email  
"email@domain.com"
```

```
git config --global user.name  
"Your Name"
```

# Let's get started!

- Go to [github.com/r-academy/tidyds](https://github.com/r-academy/tidyds)
- Hit ‘Fork’ button
- Click ‘Clone or Download’, copy the URL
- In RStudio, File – New Project – Version Control – Git. Paste URL
- In File pane (bottom-right), click vignettes-‘`m1_essentials.Rmd`’ to open it

# Working directory

```
> fs::dir_tree()  
.  
├── 003_kamisdata_Debat-Pilpres1-2019.Rproj  
├── Dockerfile  
├── R  
│   └── cari.R  
│   └── impor.R  
├── README.md  
└── data  
    └── debat-pilpres1-2019.rda  
└── data-raw  
    └── debat_pilpres1_2019.R  
└── install.R  
└── vignettes  
    ├── aswansyahputra-frekuensidansentimen.Rmd  
    ├── aswansyahputra-frekuensidansentimen.html  
    └── aswansyahputra-frekuensidansentimen_files  
> |
```

## 3 principles for naming files:

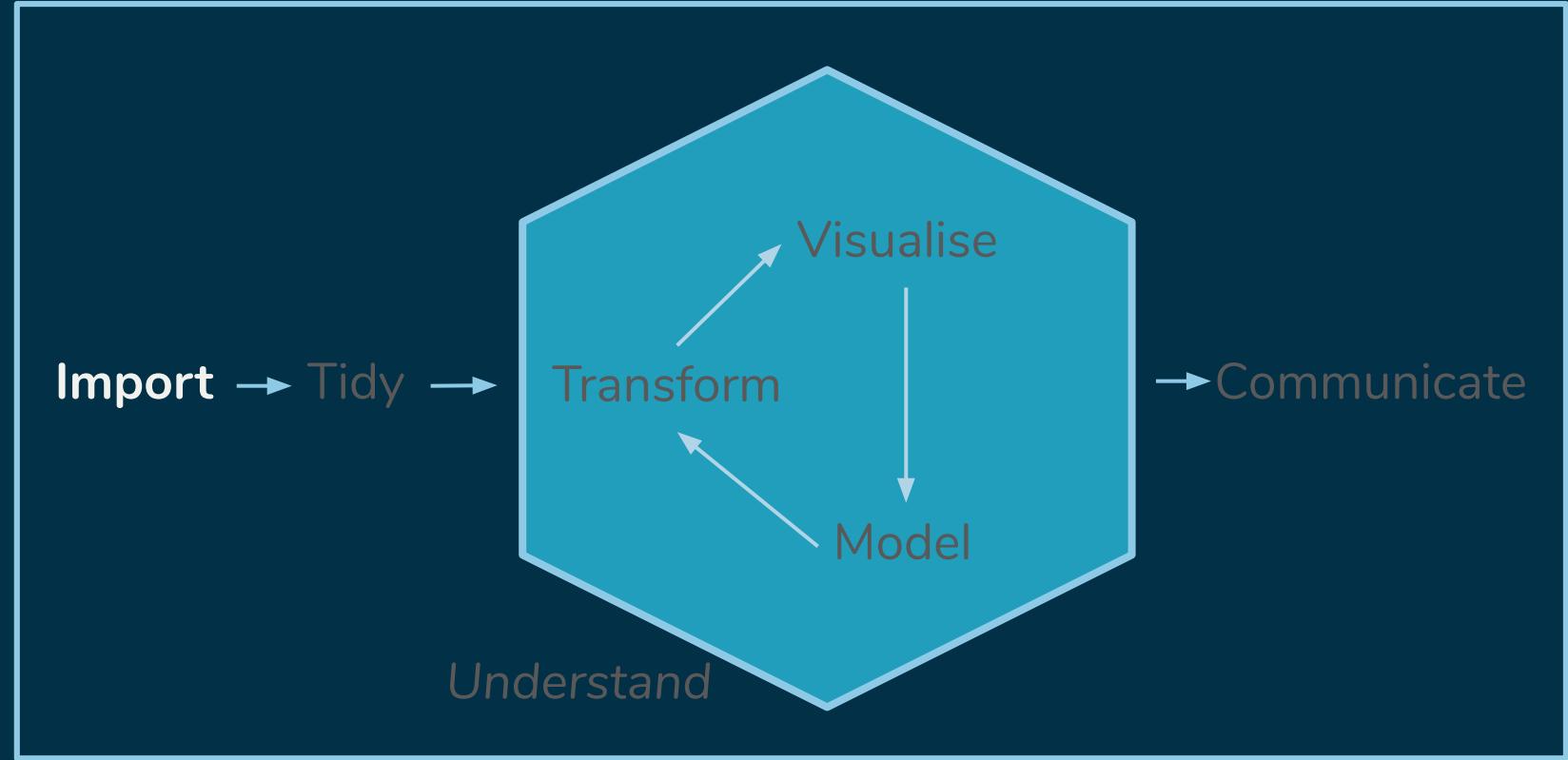
- Machine readable
- Human readable
- Default ordering

## More info:

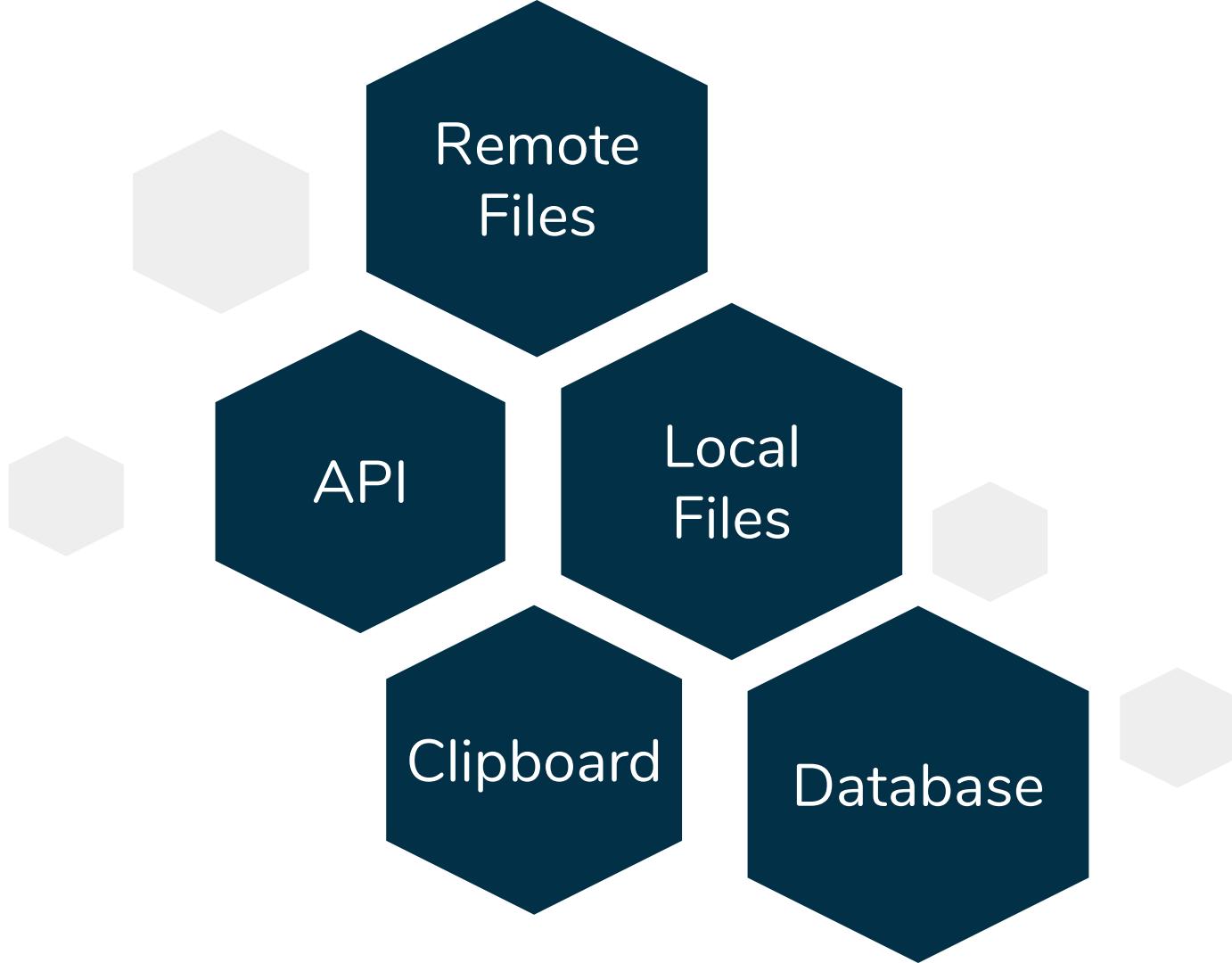
[speakerdeck.com/jennybc/  
how-to-name-files](https://speakerdeck.com/jennybc/how-to-name-files)

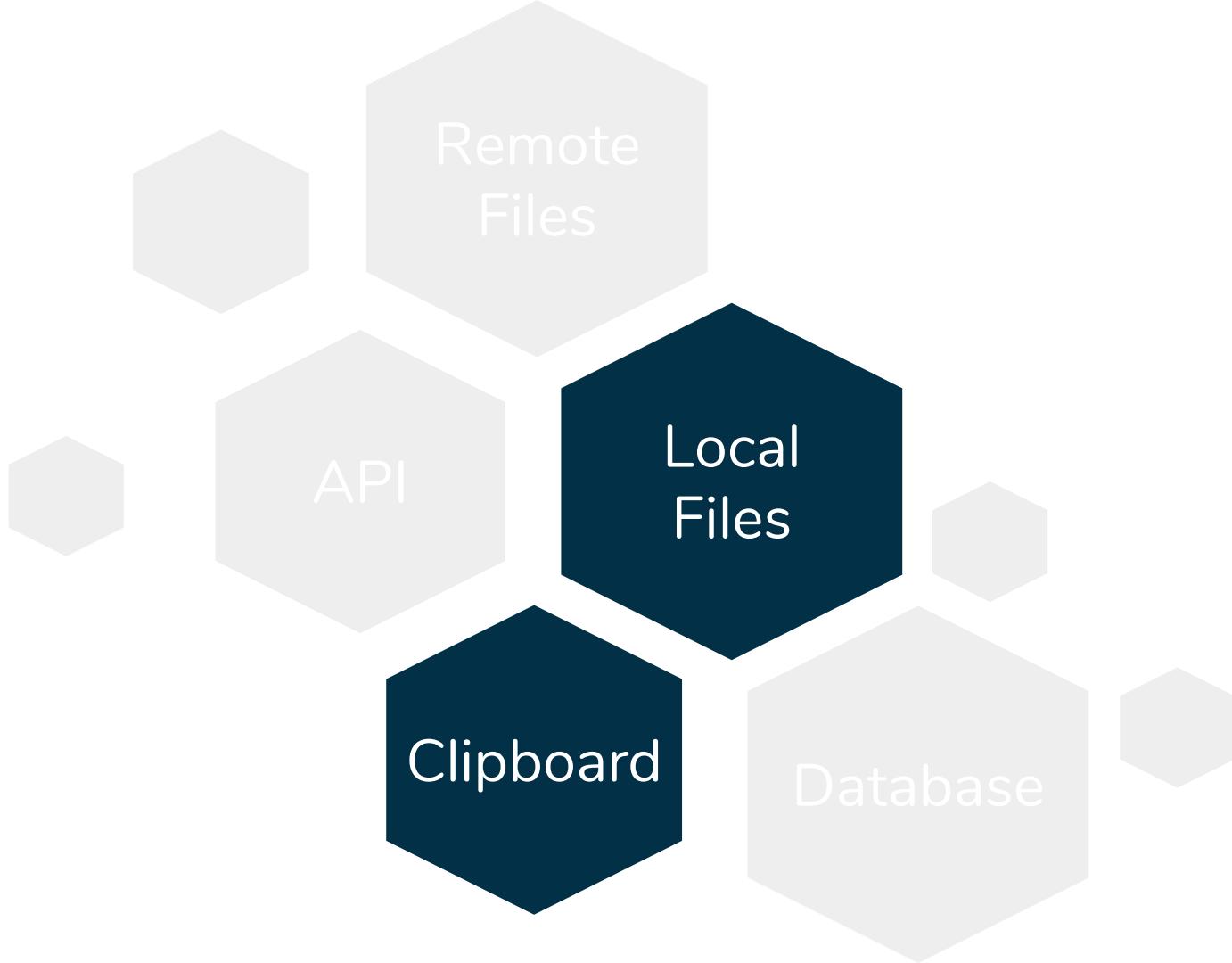
# How to use git?

- In Environment Pane, hit ‘Git’ tab
- Click commit, a window pane will appear
- Select all files (Ctrl + A), click ‘Stage’
- Fill commit message, then click ‘Commit’
- Hit ‘Push’ Button, done!
- You may check your GitHub now!



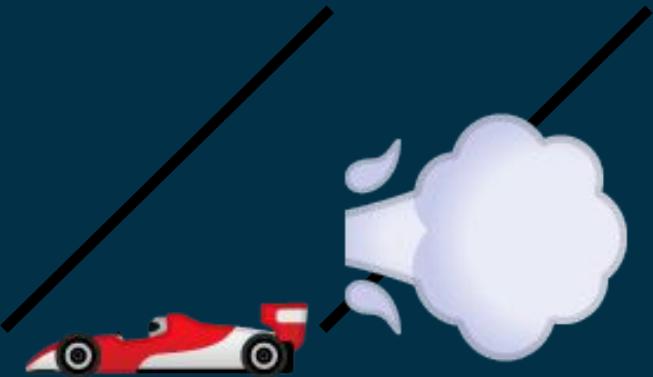
*Program*





- `vroom()`: comma separated (CSV) files
- `vroom()`: tab separated files
- `vroom()`: general delimited files
- `vroom()`: fixed width files
- `vroom()`: tabular files where columns are separated by white-space.

# vroom



# Tame data is lovely

- Use syntactical names for column names
- Use consistent case: snake\_case, camelCase, caterpillar.case
- Cast column type accordingly: <chr>, <dbl>, <lgl>, <date>, etc
- Treat <fct> carefully!
- Preferably turn implicit missing observation into explicit NA value

```
x <- something  
for (i in seq_along(x) {  
  function(x[[i]])  
}
```

```
x <- something  
*apply(x, function)
```

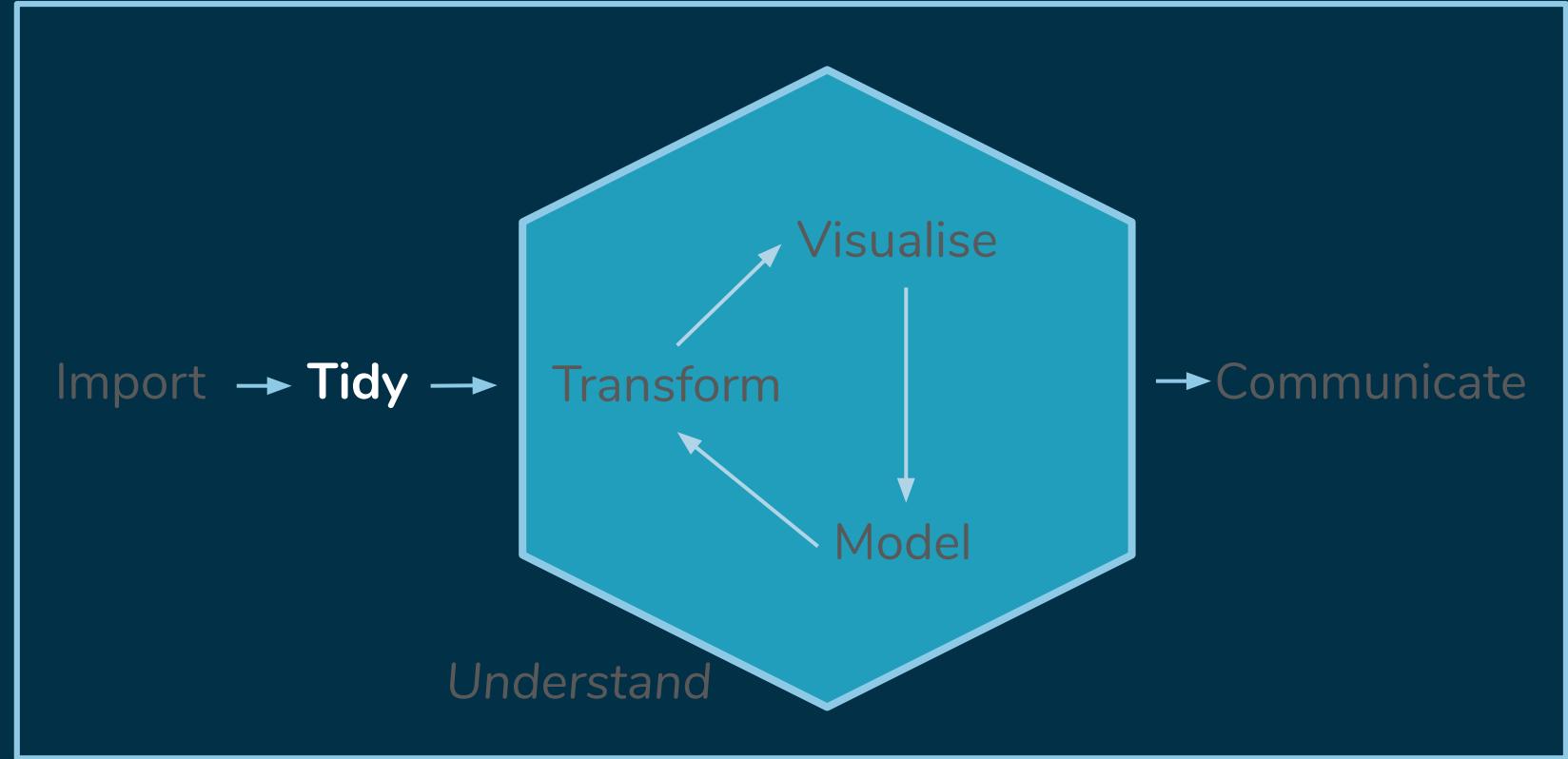
```
x <- something  
map_*(x, function)
```

# purrr



# Let's do practice!

- Open ‘m2\_data-import.Rmd’
- Do not forget to push your works into GitHub!



*Program*

Tidy datasets are all alike,  
but every messy dataset is  
messy in its own way!

- Hadley Wickham

# A Tidy dataset

	Name	Gender	Age
1	Jane	Male	44
2	Lisbon	Female	38
3	John	Male	NA

There  
Each  
Each  
Each

# A variable has its own column

	Var. 1	Var. 2	Var. 3
Obs. 1	A	B	C
Obs. 2	D	E	F
Obs. 3	G	H	I

There  
Each  
Each  
Each

# An observation has its own row

	Var. 1	Var. 2	Var. 3
Obs. 1	A	B	C
Obs. 2	D	E	F
Obs. 3	G	H	I

There  
Each  
Each  
Each

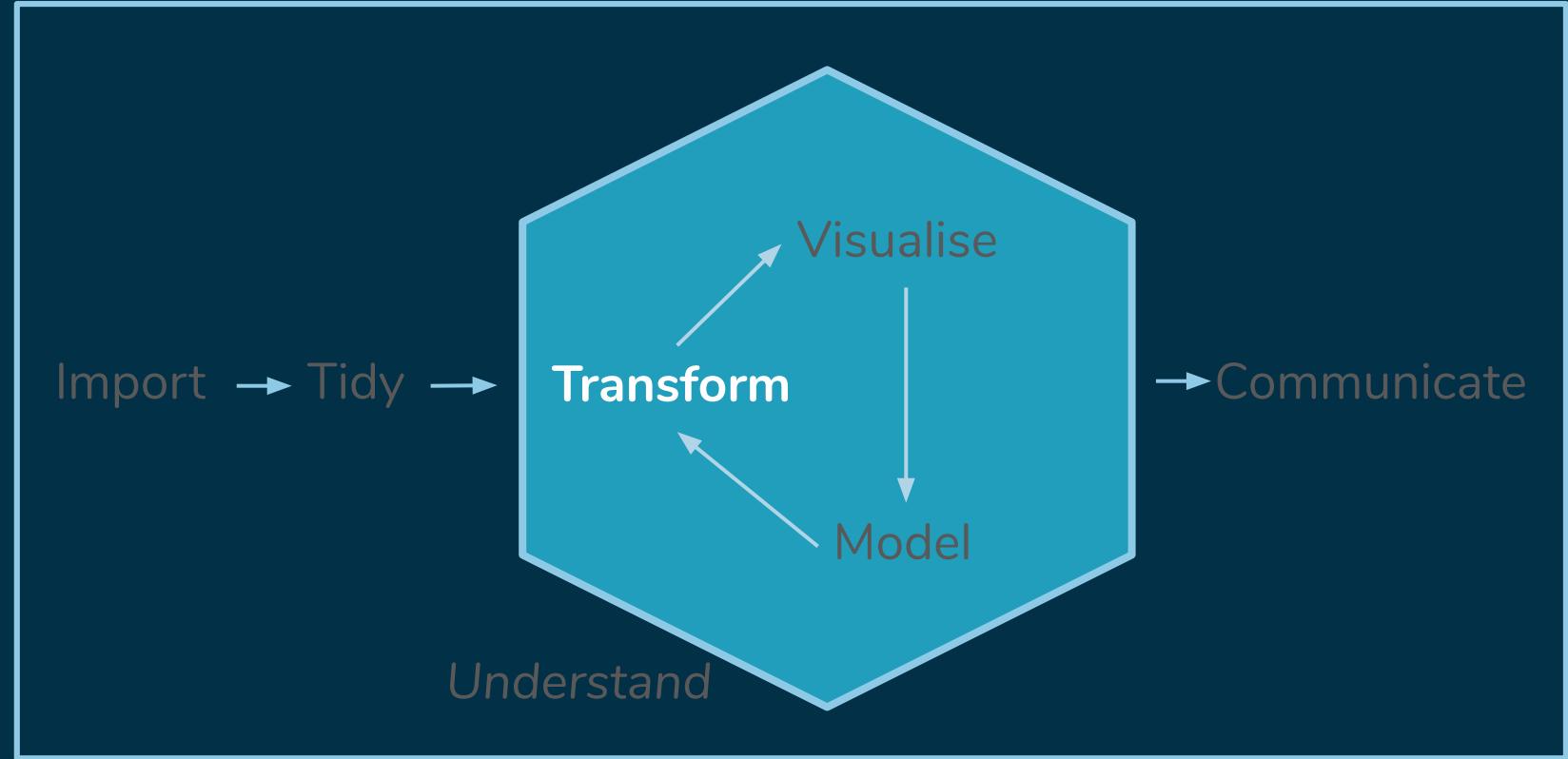
# A value has its own cell

	Var. 1	Var. 2	Var. 3
Obs. 1	A	B	C
Obs. 2	D	E	F
Obs. 3	G	H	I

There  
Each  
Each  
Each



Artwork by @allison\_horst



Program

dplyr : go wrangling



## dplyr basic functions:

- `filter()` selects rows based on their values
- `mutate()` creates new variables
- `select()` picks columns by name
- `summarise()` calculates summary statistics
- `arrange()` sorts the rows

## tidyverse basic functions:

- `gather()` wide-format >> long-format
- `spread()` long-format >> wide-format
- `fill()` fills value based on previous entry
- `complete()` turns implicit missing values into explicit

## Operators:

- `!` (not)
- `|` (or)
- `&` (and)
- `==, !=`
- `<, <=, >, >=`
- `%in%`
- `is.na()`

How can I  
chain?

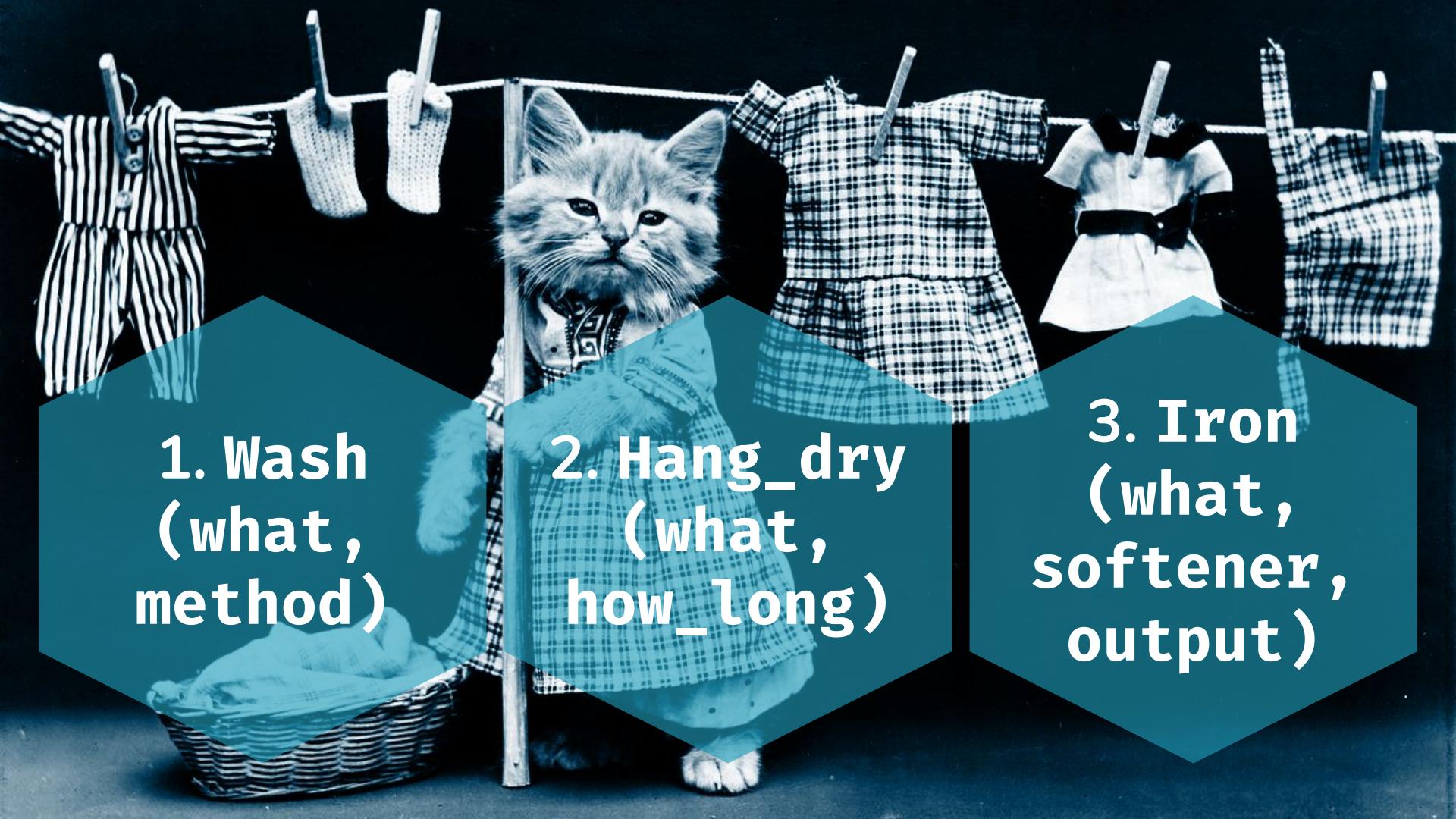




1. Wash

2. Hang\_dry

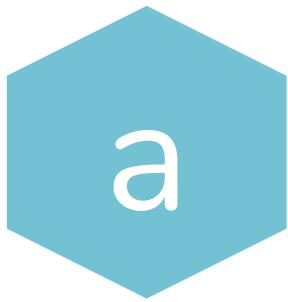
3. Iron



**1. Wash  
(what,  
method)**

**2. Hang\_dry  
(what,  
how\_long)**

**3. Iron  
(what,  
softener,  
output)**



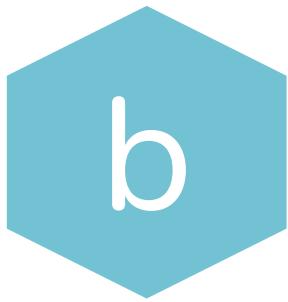
a

```
> washed_clothes <- wash(what = dirty_clothes,  
                           method = "handwash")  
> dry_clothes <- hang_dry(what =  
                           washed_clothes,  
                           how_long = 8)  
> iron(what = dry_clothes, softener = TRUE,  
        output = "neat.clothes")
```



a

```
> washed_clothes <- wash(what = dirty_clothes,  
                           method = "handwash")  
> dry_clothes <- hang_dry(what =  
                           washed_clothes,  
                           how_long = 8)  
> iron(what = dry_clothes, softener = TRUE,  
        output = "neat.clothes")
```



b

```
> iron(  
  hang_dry(  
    wash(what = dirty_clothes, method =  
          "handwash"),  
    how_long = 8),  
  softener = TRUE,  
  output = "neat.clothes")
```

```
function(arg1, arg2, arg3, ...)
```

```
arg1 %>%  
  function(arg2, arg3, ...)
```

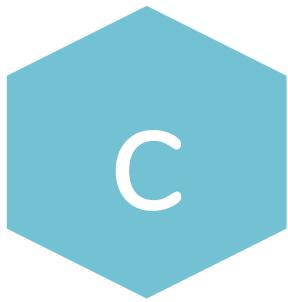
---

```
function(arg1, arg2, arg3, ...)
```

```
arg2 %>%  
  function(arg1, arg2=., arg3,  
          ...)
```

# magrittr





C

```
> wash(what = dirty_clothes,  
       method = "handwash") %>%  
  hang_dry(how_long = 8) %>%  
  iron(softener = TRUE,  
        output = "neat.clothes")
```

# R Syntax Comparison

## Dollar sign

`goal(data$x, data$y)`

- A.k.a base syntax
- Subsetting data by using ‘`[]`’

## Formula

`goal(y~x, data=data)`

Mostly used in modeling and statistical test

## Tidyverse

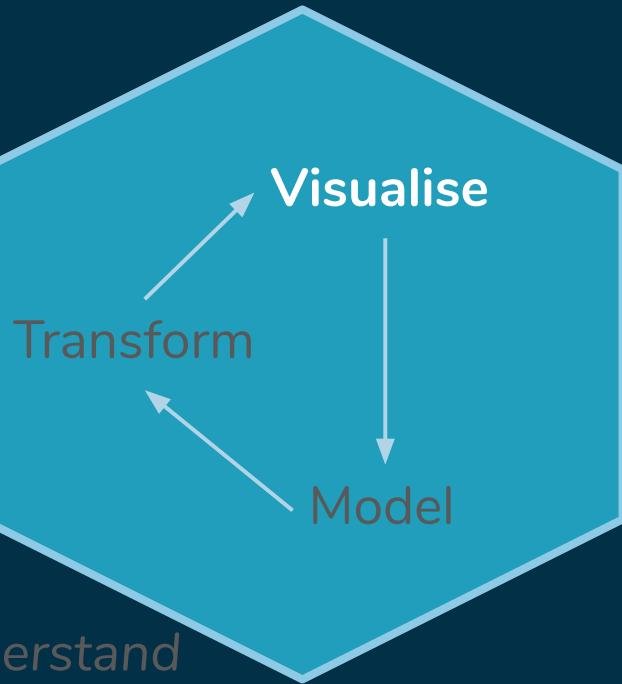
`data %>% goal(x, y)`

- Expecting data as the first argument
- Plotting using ‘`+`’ flavour

# Let's do practice!

- Open ‘m3\_tidy-data-carpentry.Rmd’
- Do not forget to push your works into GitHub!

Import → Tidy



→ Communicate

*Program*

# ggplot2: VISUAL DATA EXPLORATION



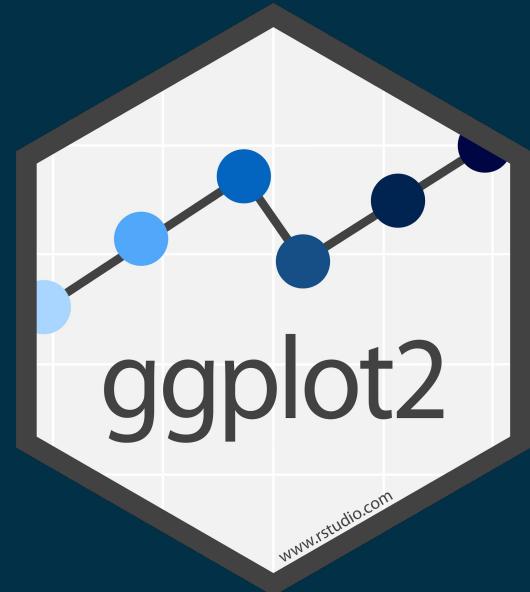
Artwork by @allison\_horst

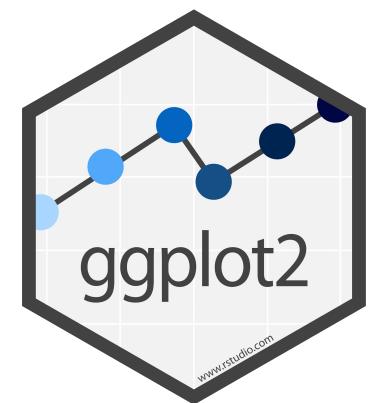
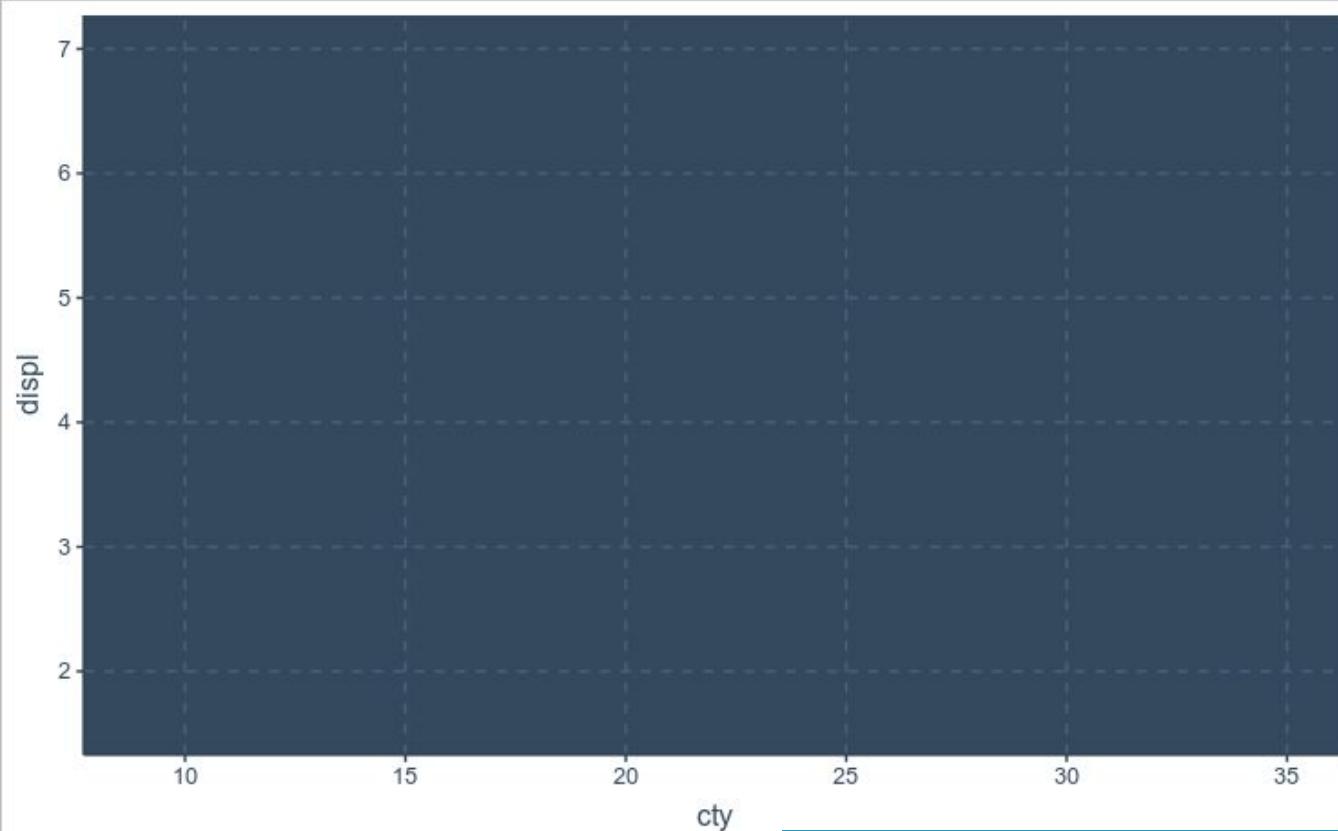
```
ggplot(data) +  
  geom_X(mapping=aes(...)) +  
  ...  
  
ggplot(data, mapping=aes(...)) +  
  geom_X() +  
  ...
```

---

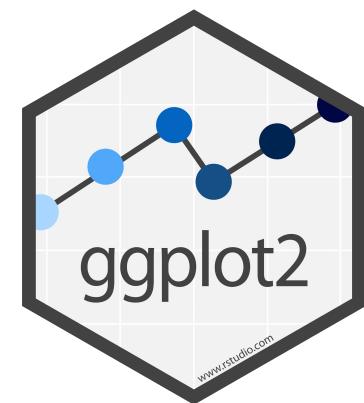
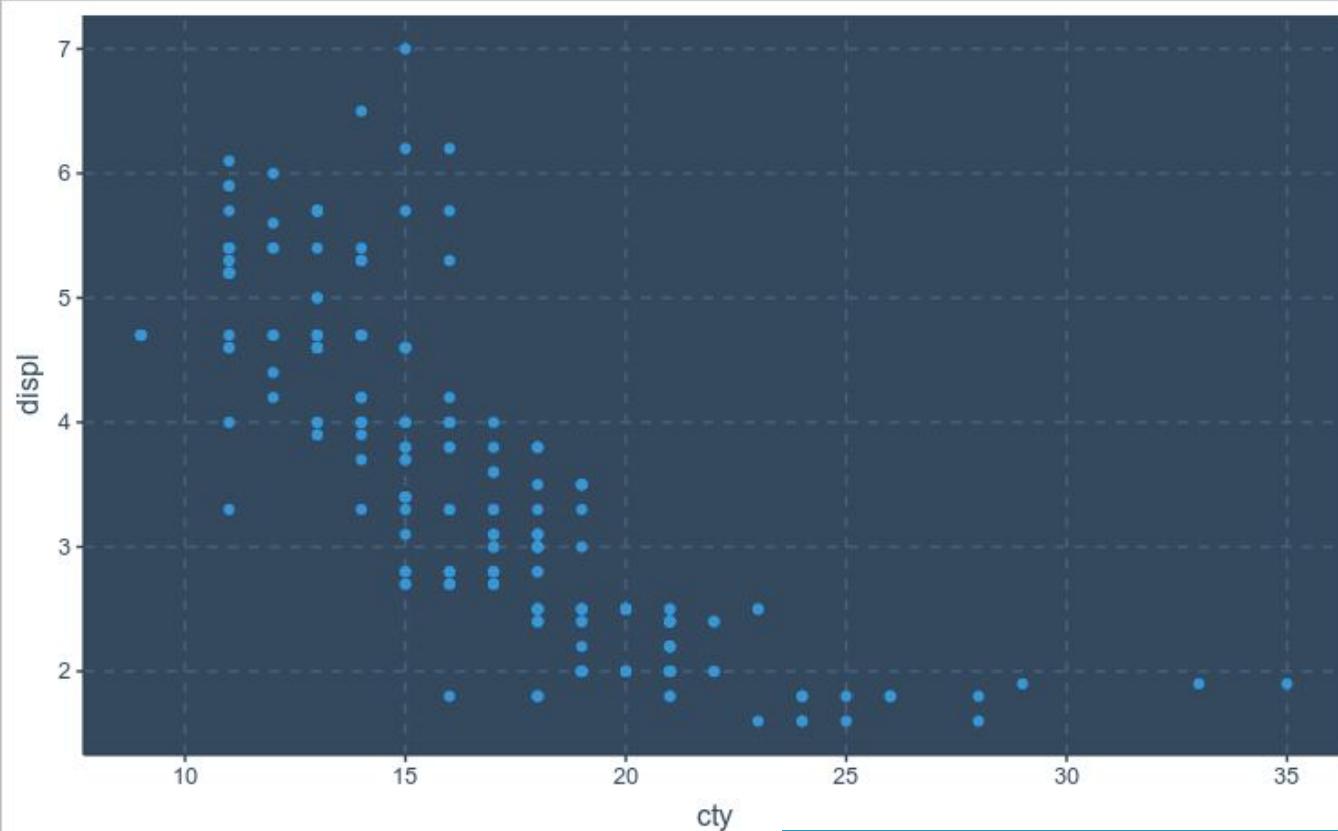
```
data %>%  
  ggplot(mapping=aes(...)) +  
  geom_X() +  
  ...
```

# ggplot2

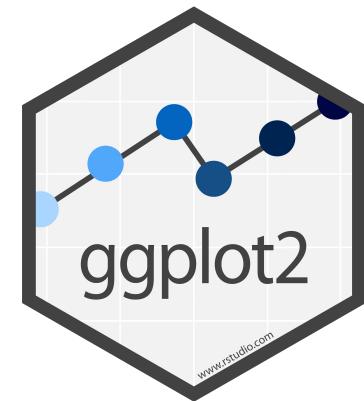
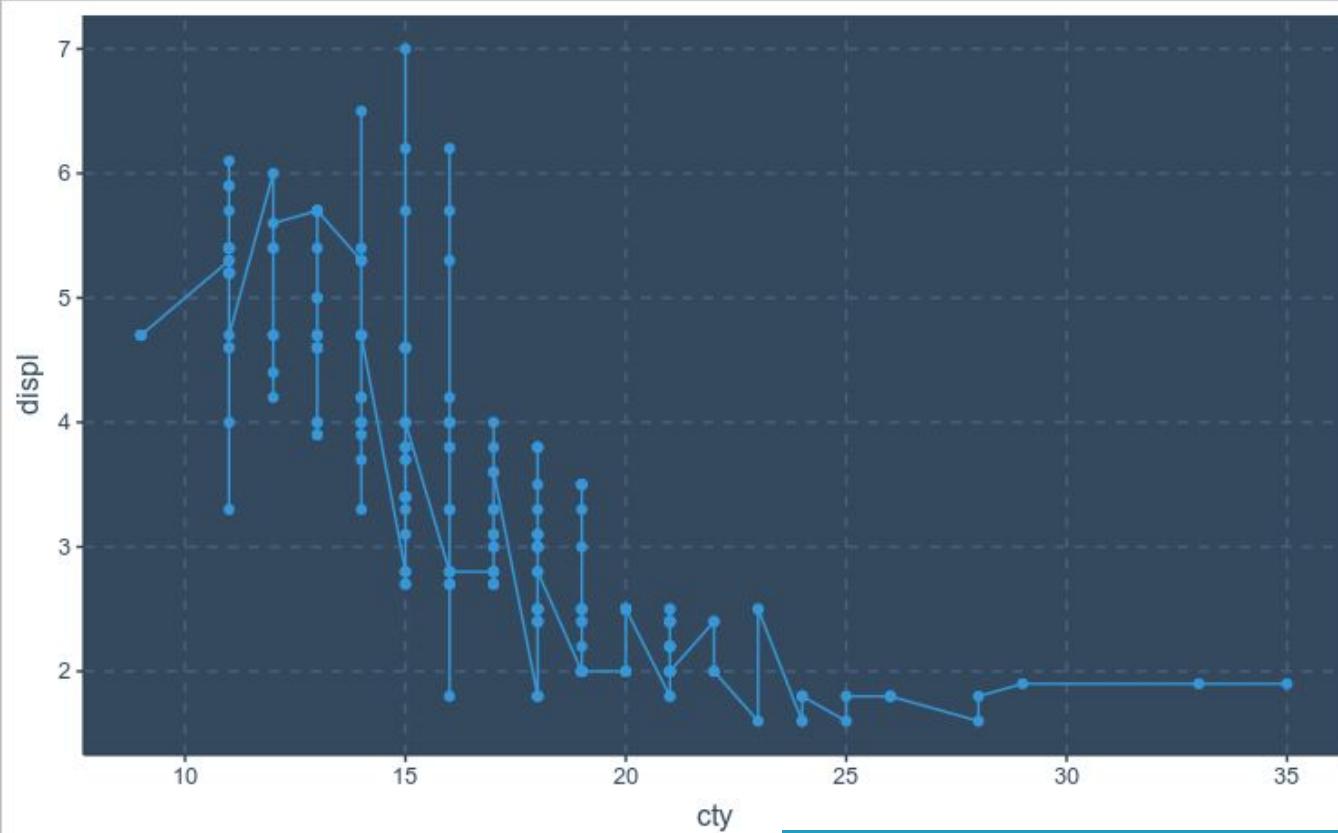




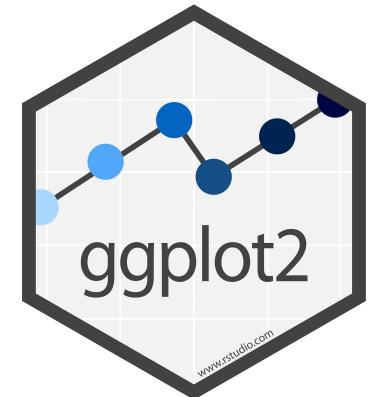
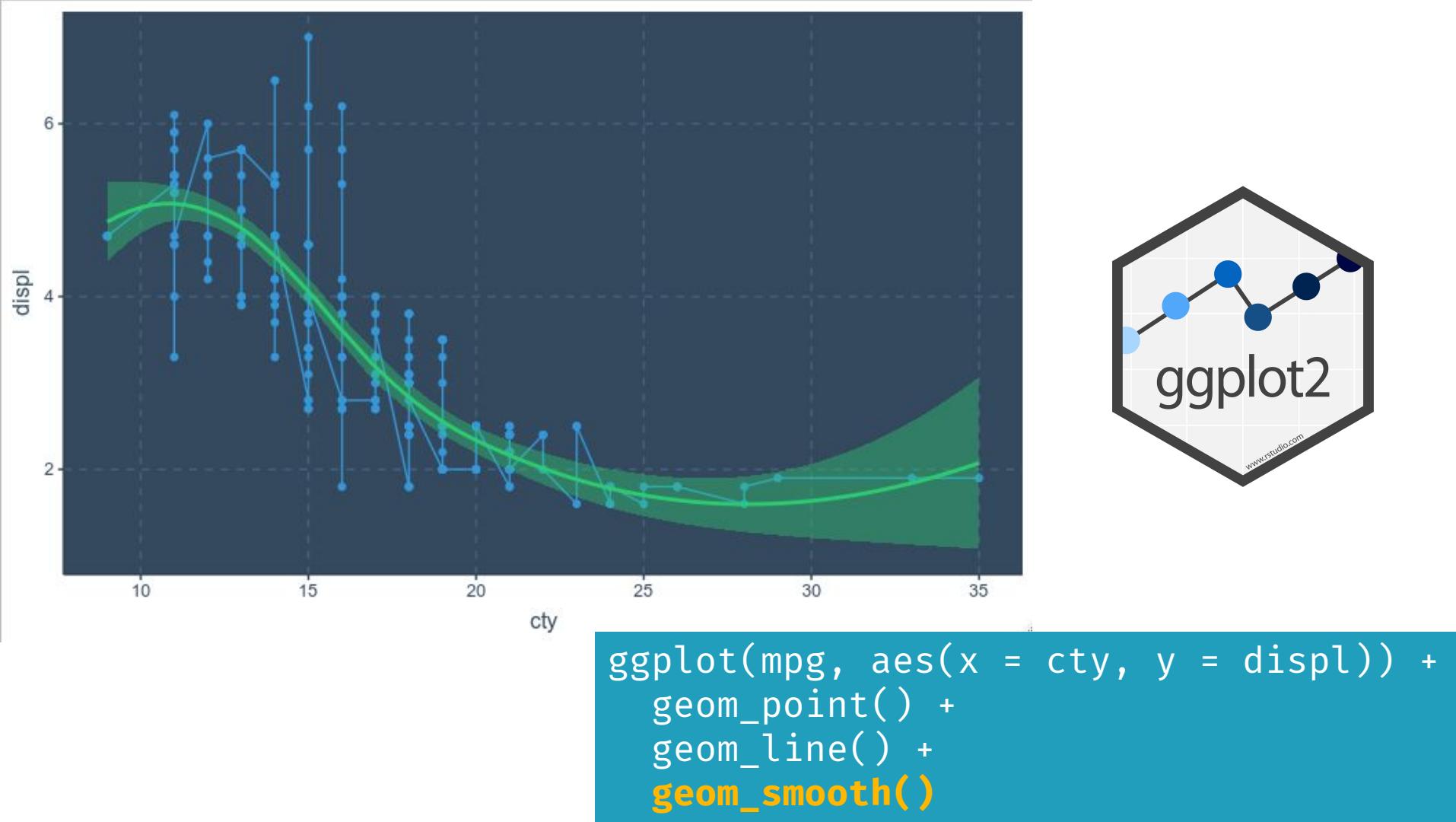
```
ggplot(mpg, aes(x = cty, y = displ))
```



```
ggplot(mpg, aes(x = cty, y = displ)) +  
  geom_point()
```

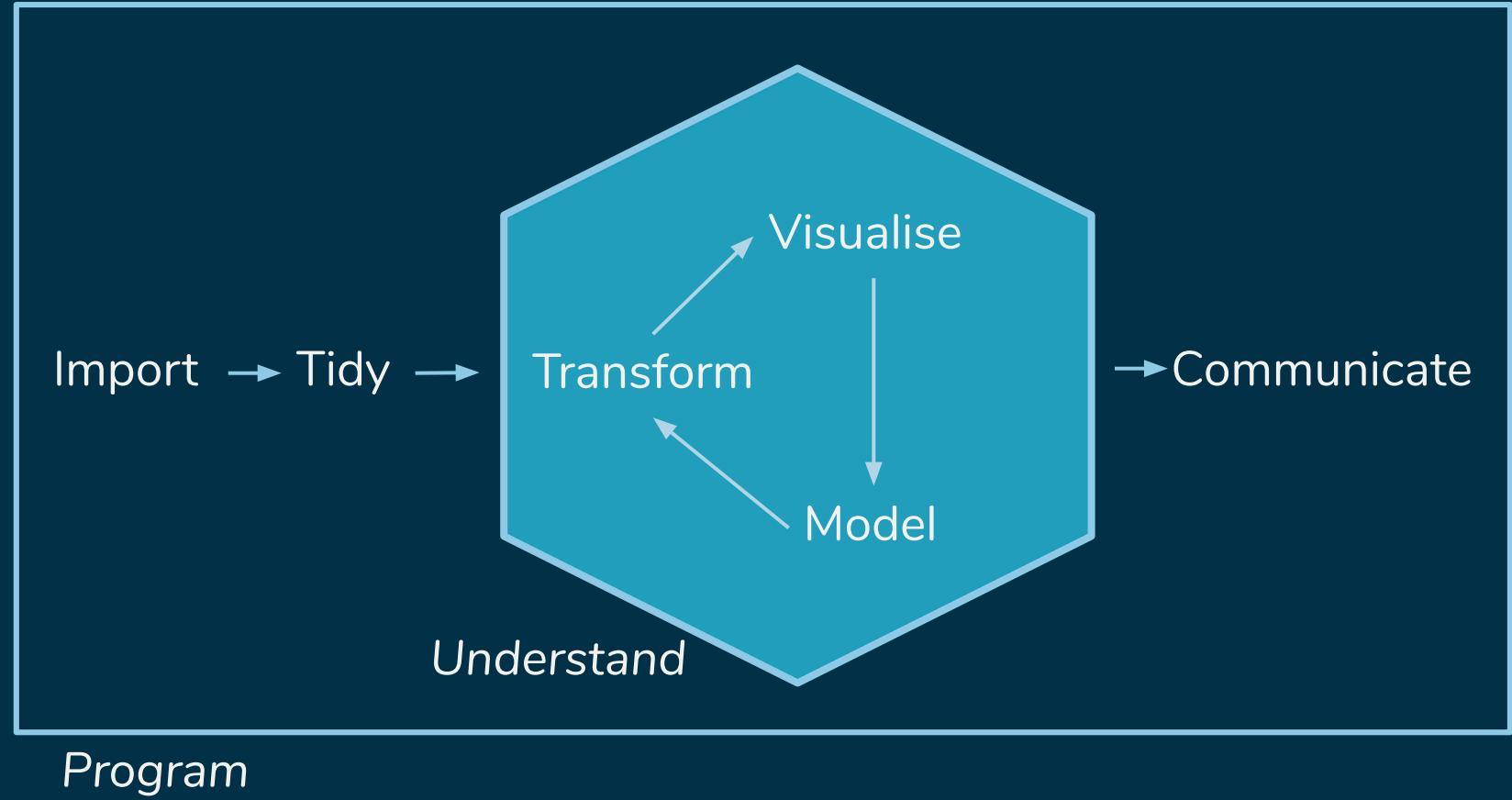


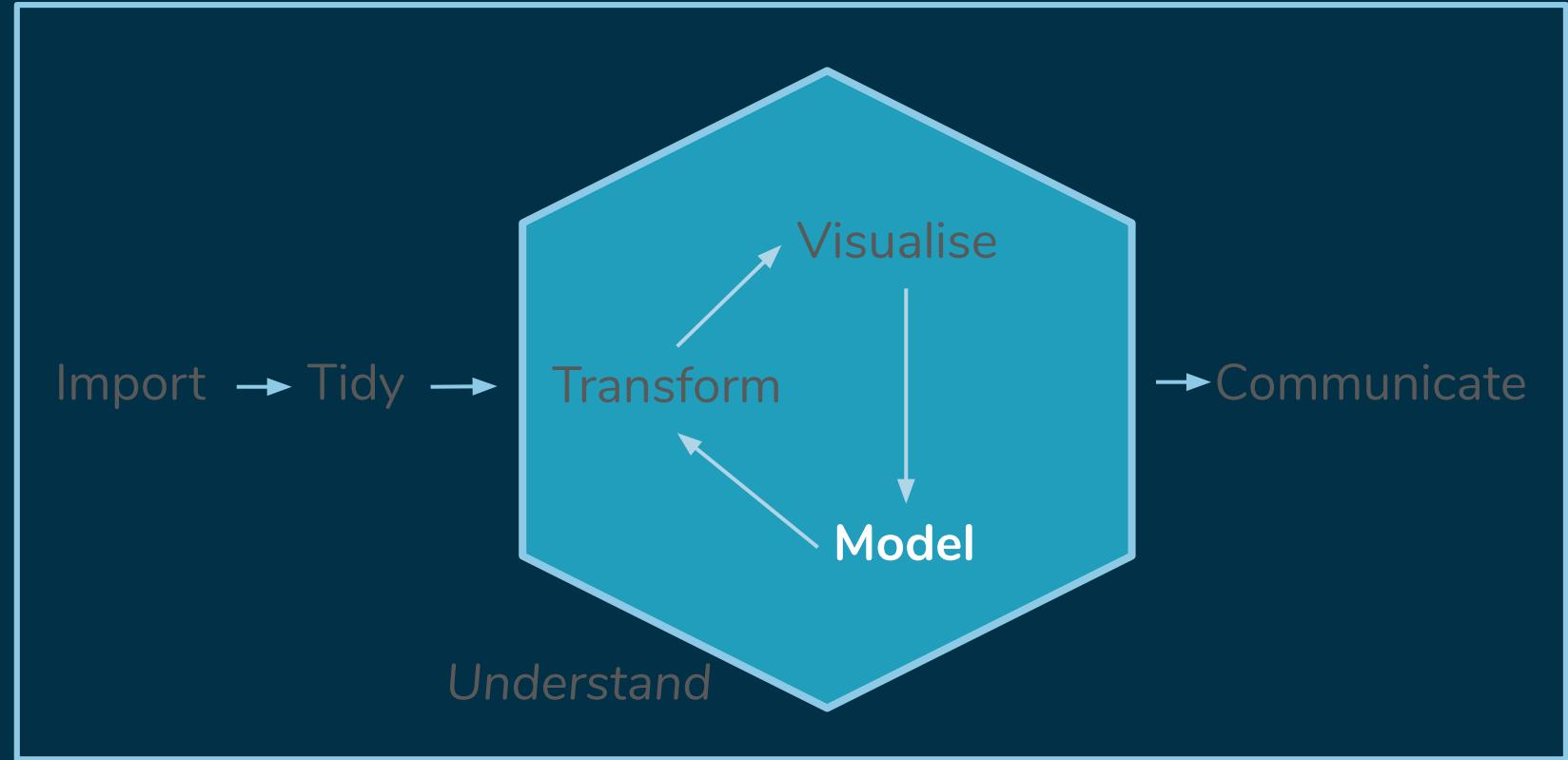
```
ggplot(mpg, aes(x = cty, y = displ)) +  
  geom_point() +  
  geom_line()
```



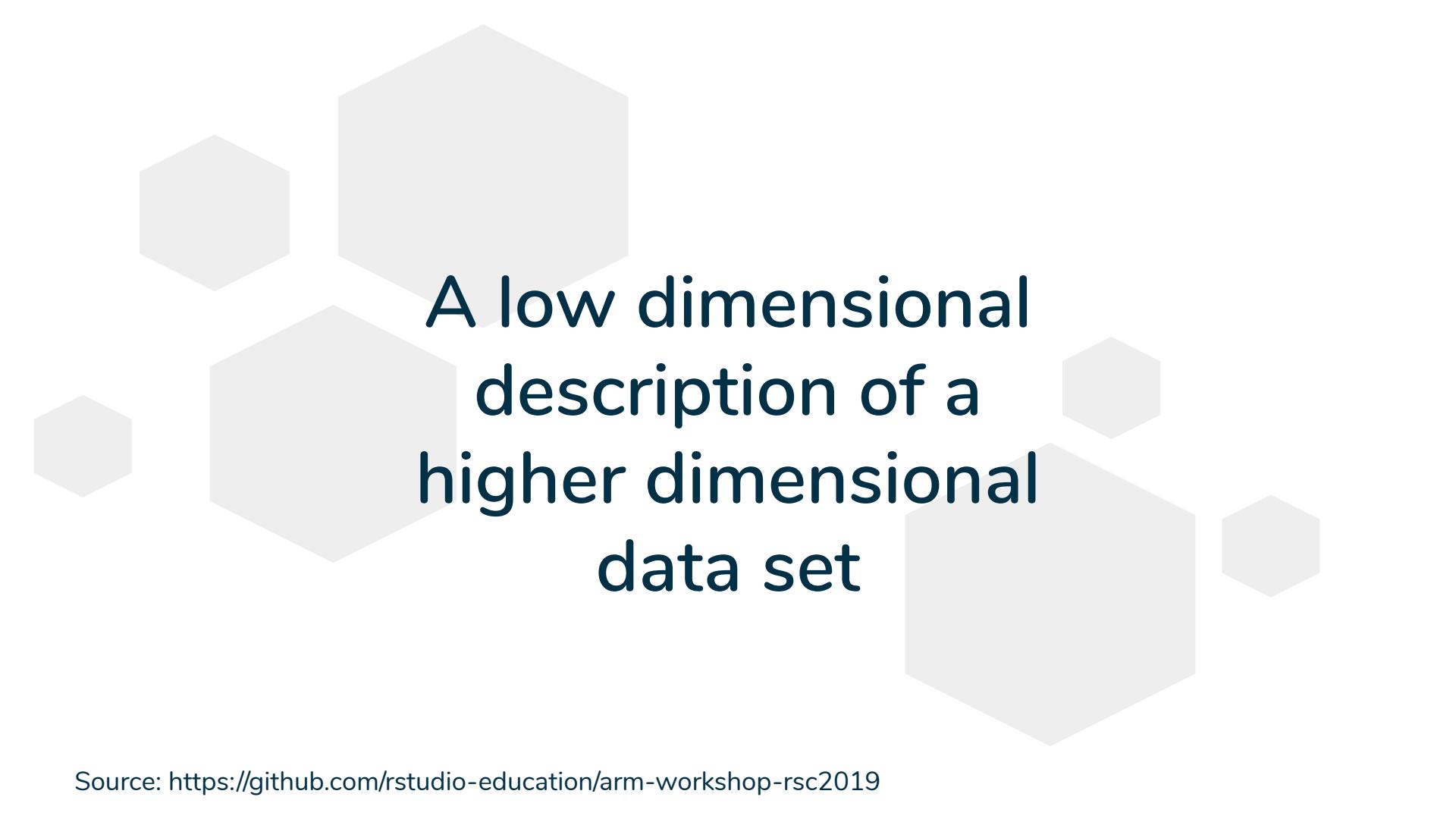
# Let's do practice!

- Open ‘m4\_data-visualisation.Rmd’
- Do not forget to push your works into GitHub!

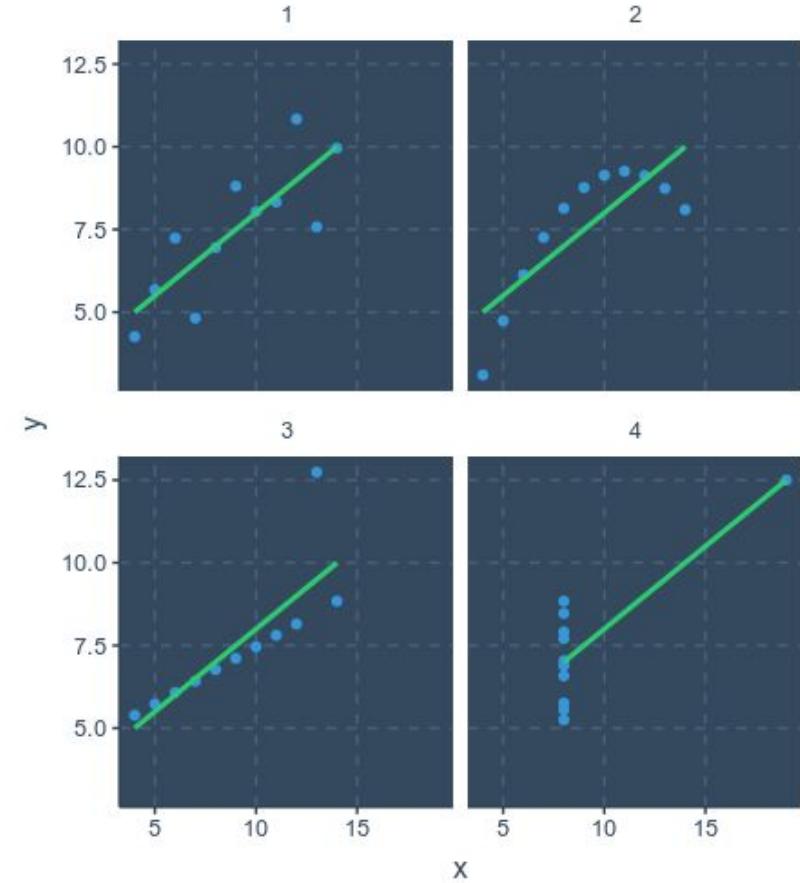




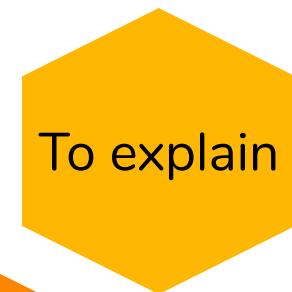
*Program*



A low dimensional  
description of a  
higher dimensional  
data set

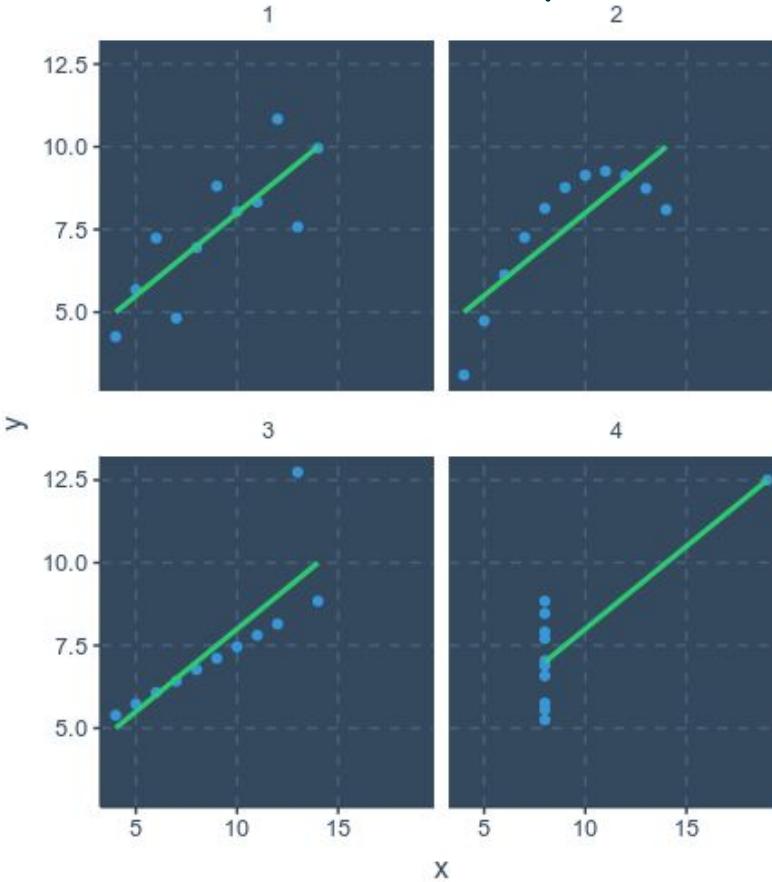


Outcome ~ Predictor/Explanatory



- All models are wrong, but some are useful – George Box

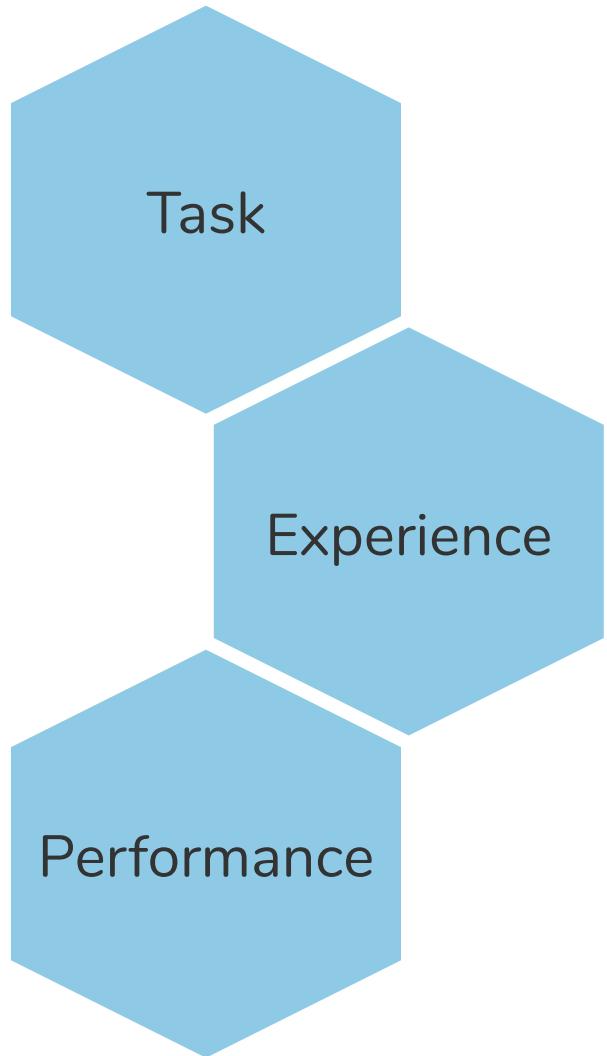
# Anscombe's Quartet



Mean of x	9	exact
Sample variance of x	11	exact
Mean of y	7.50	to 2 decimal places
Sample variance of y	4.125	$\pm 0.003$
Correlation between x and y	0.816	to 3 decimal places
Linear regression line	$y = 3.00 + 0.500x$	to 2 and 3 decimal places, respectively
$R^2$	0.67	to 3 decimal places

# Machine Learning

$$y = f(x) + e$$



# Supervised learning

# Unsupervised learning

Springer Texts in Statistics

Gareth James  
Daniela Witten  
Trevor Hastie  
Robert Tibshirani

An Introduction  
to Statistical  
Learning

with Applications in R

 Springer

Pre-process

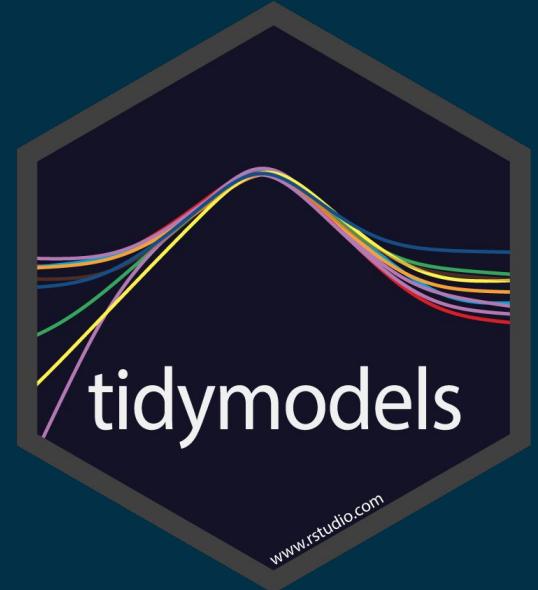


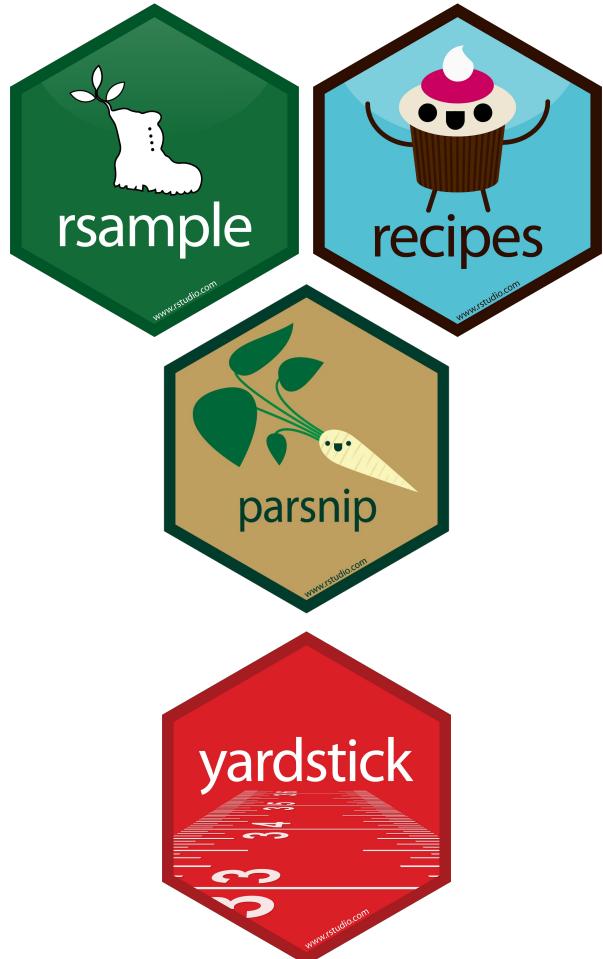
Train



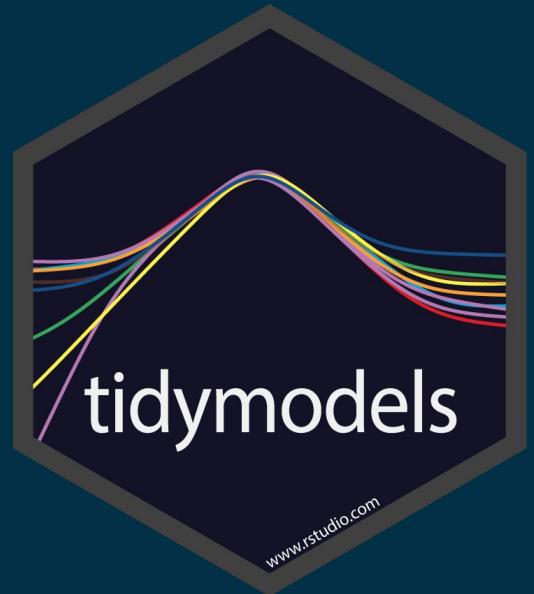
Validate

tidymodels





# tidymodels

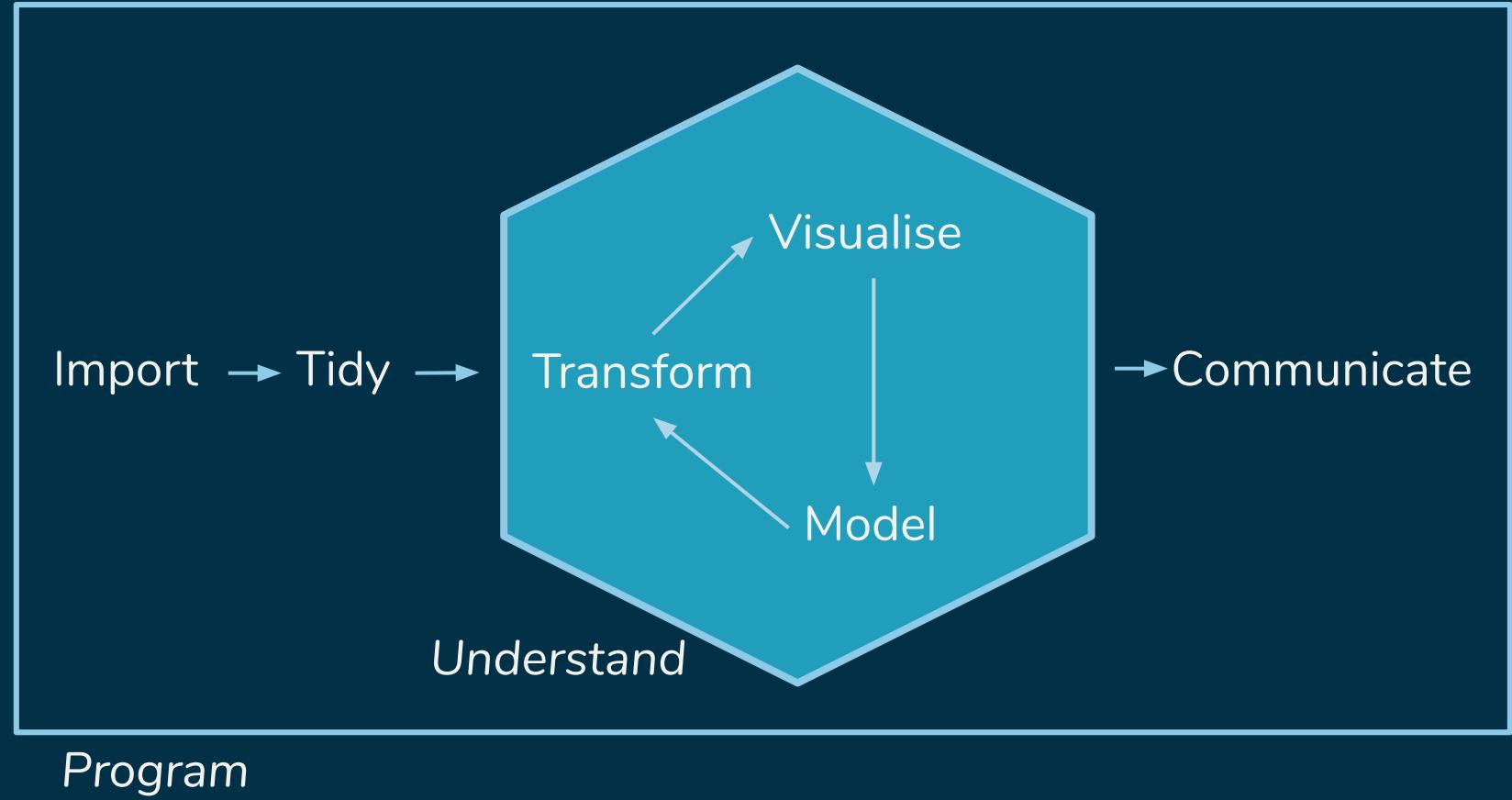


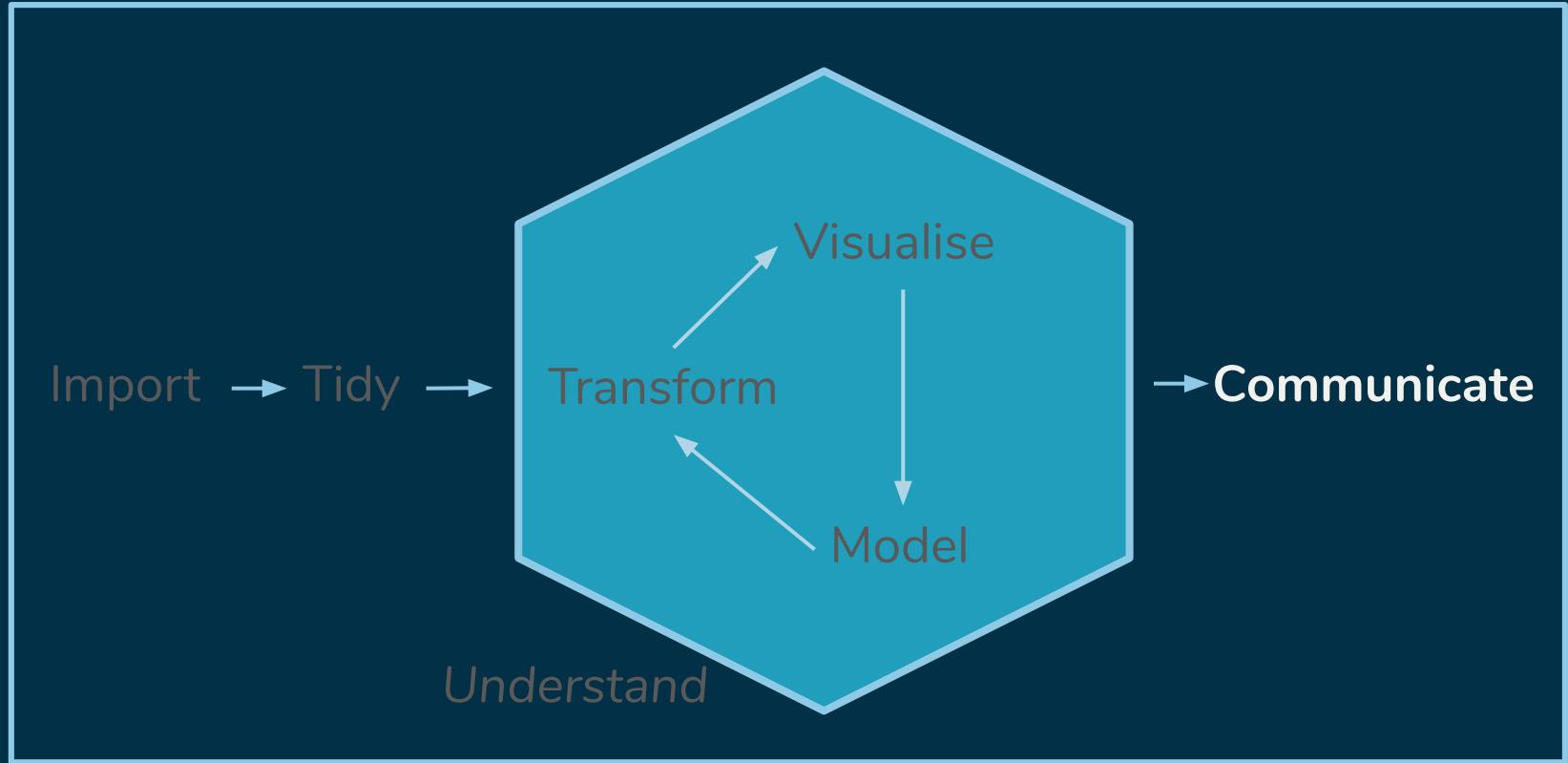
# Let's do practice!

- Open ‘m5\_model-1.Rmd’
- Do not forget to push your works into GitHub!

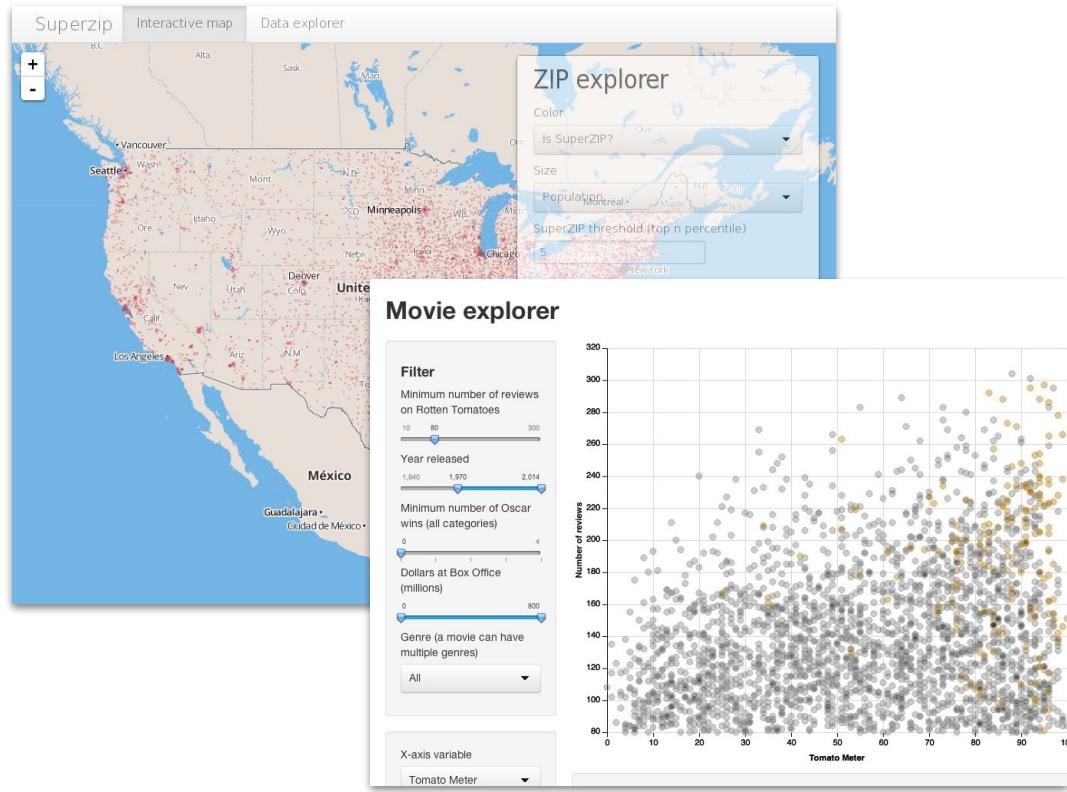
# Let's do practice!

- Open ‘m6\_model-2.Rmd’
- Do not forget to push your works into GitHub!





*Program*



Images from [shiny.rstudio.com](http://shiny.rstudio.com)



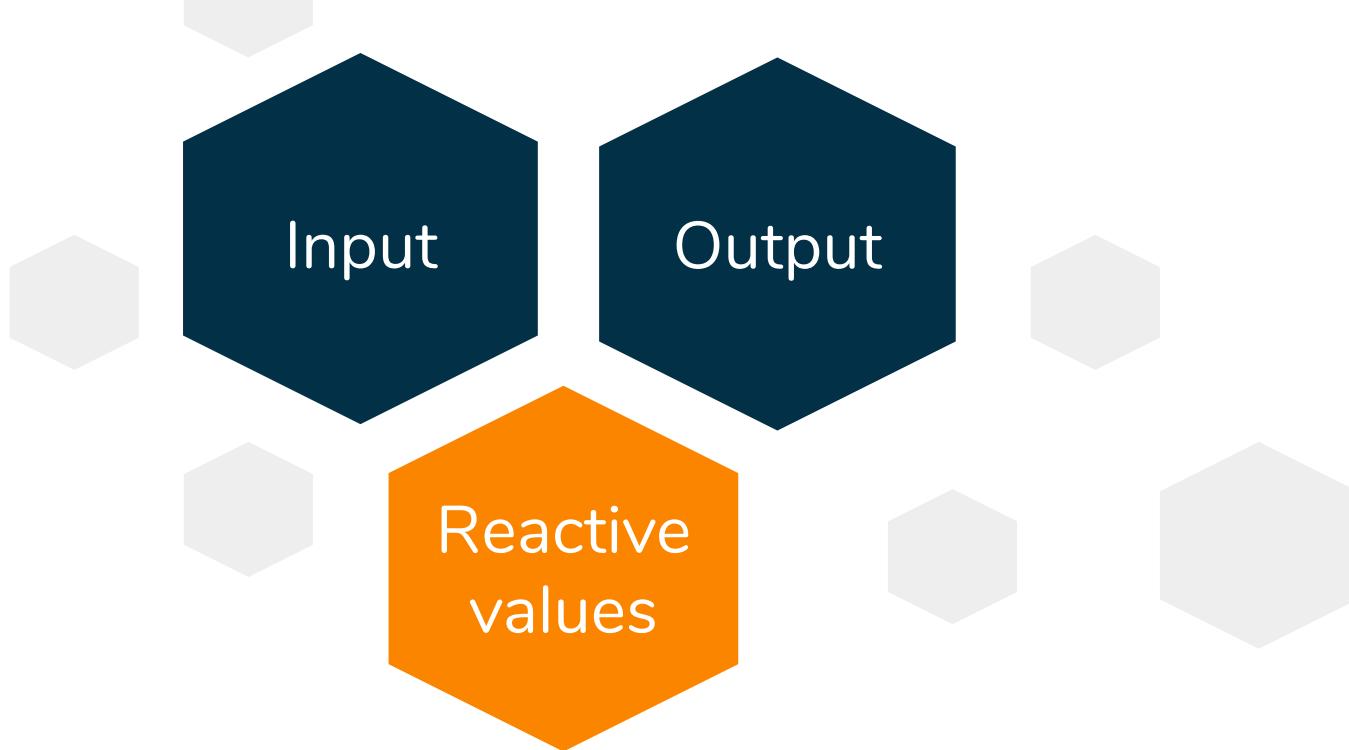
```
library(shiny)
mydata <- load("mydata.rda")

ui <- fluidPage(
  # define user interface here
)

server <- function(input, output) {
  # put logic and computation here
}

shinyApp(ui = ui, server = server)
```

# Important terms!



```
library(shiny)
mydata <- load("mydata.rda")

ui <- fluidPage(
  # define user interface here
)

server <- function(input, output) {
  # put logic and computation here
}

shinyApp(ui = ui, server = server)
```

```
library(shiny)  
mydata <- load("mydata.rda")
```

- Inputs are defined and laid out
- Outputs are laid out

```
ui <- fluidPage(  
  # define user interface here  
)
```

- Inputs are taken
- Outputs are calculated/made

```
server <- function(input, output) {  
  # put logic and computation here  
}
```

```
shinyApp(ui = ui, server = server)
```

1. Take input value from `input$xx`
2. Perform processing/calculation in reactive context (expression/observer)
3. Build objects to display using `render*()`
4. Save objects to display to `output$xx`

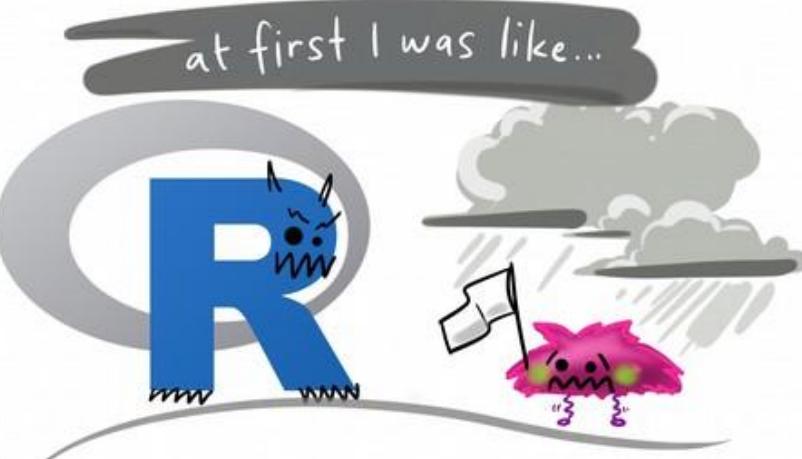
# Let's do practice!

- Open ‘m7\_shiny-1.Rmd’
- Do not forget to push your works into GitHub!

# Let's do practice!

- Open ‘m8\_shiny-2.Rmd’
- Do not forget to push your works into GitHub!

# Congrats!



...but now it's like...



Artwork by @allison\_horst

# Contact me

Name: Muhammad Aswan Syahputra  
Email: [aswansyahputra@sensolution.id](mailto:aswansyahputra@sensolution.id)  
Phone: +62 822 3465 3816  
Twitter: [@aswansyahputra\\_](https://twitter.com/aswansyahputra_)